Geocortex Viewer for HTML5 2.5

Administrator and Developer Guide

Geocortex® | by Latitude Geographics®

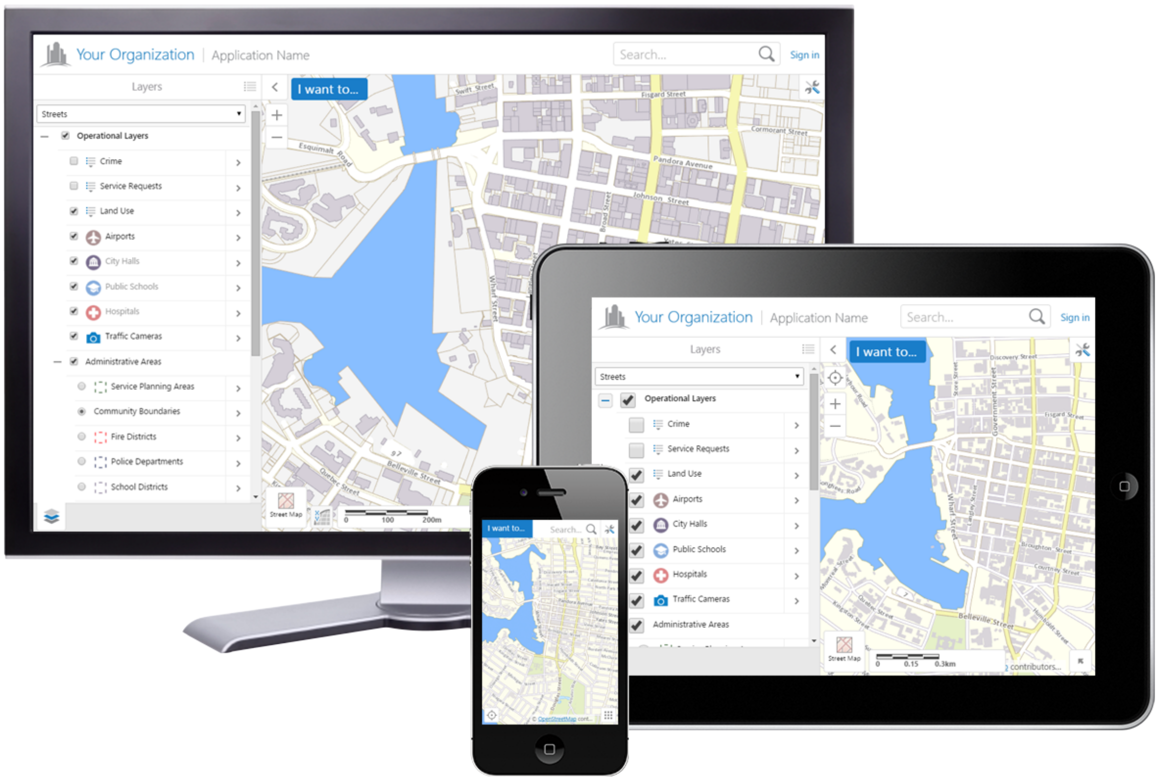
Contents

1 Welcome

Welcome to the Geocortex Viewer for HTML5 2.5 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 JavaScript API and HTML5 Viewer Support Forum: http://support.geocortex.com/Forums/ForumView.aspx?pageid=0&mid=2&ItemID=9
- ESRI ArcGIS API for JavaScript: http://help.arcgis.com/en/webapi/javascript/arcgis/help/jsapi_start.htm

2.2 Email Support

If you have a question that is not suitable for posting to the forum, and you have an active Support agreement, send an email message to the Geocortex Support team at `support@geocortex.com`. The Support team will respond by email.

2.3 Viewer Log

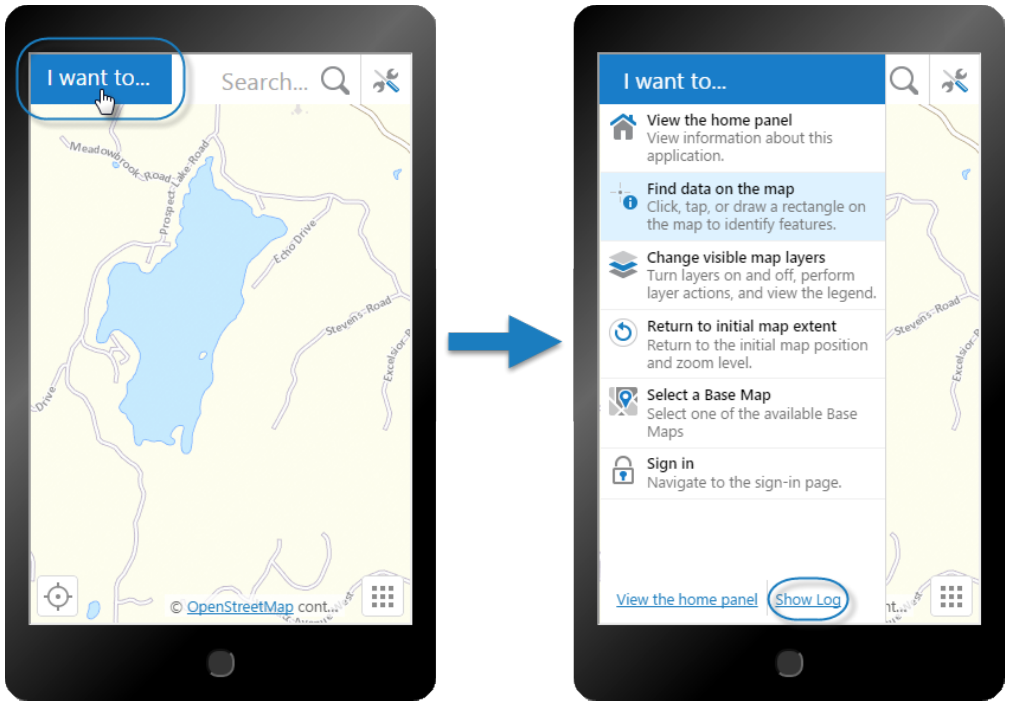
The Viewer keeps a log of its activities. This log is helpful when tracking down incompatibilities and errors. Attaching relevant log entries to a support forum post or email can be very useful.

To view the log file:

- **On a desktop PC or tablet:** Press **Ctrl** + **Shift** + **~**.

  Alternatively, hold down the **Shift** key and press **Esc** twice.

- **On a handheld device:** Tap the **I Want To** menu button, and tap **Show Log**.


I want to...
Search...
Meadowbrook Road
Lake Road
Prospect
Echo Drive
Stevens Road
Excelsior
Drive
© OpenStreetMap cont...
I want to...
View the home panel
View information about this application.
Find data on the map
Click, tap, or draw a rectangle on the map to identify features.
Change visible map layers
Turn layers on and off, perform layer actions, and view the legend.
Return to initial map extent
Return to the initial map position and zoom level.
Select a Base Map
Select one of the available Base Maps
Sign in
Navigate to the sign-in page.
View the home panel
Show Log

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

2.4 Provide Feedback on Documentation

Please send comments and suggestions related to documentation to: 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. See ArcGIS API for JavaScript system requirements for more information.

The 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.

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:

- Microsoft Edge (current version preferred)
- Microsoft Internet Explorer 8.0+ (11.0 preferred)

  Support for Internet Explorer 8 is now deprecated. HTML5 Viewer 2.5 is the final version to support Internet Explorer 8. HTML5 Viewer 2.6 will require at least Internet Explorer 9.

  Internet Explorer 8 does not support geolocation or integration with third-party mapping applications.

- Mozilla Firefox (current version recommended)
- Google Chrome (current version recommended)

3.1.2 Mobile Browser Support

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

- Safari on iOS 7+
- Chrome on Android

3.2 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 **26** for information.

3.3 Geocortex Essentials Requirements

Geocortex Viewer for HTML5 2.5 works with Geocortex Essentials 4.4 and later.

3.4 ArcGIS Requirements

The Geocortex Viewer for HTML5 works with:

- ArcGIS Server 9.3
- ArcGIS Server 9.3.1
- ArcGIS Server 10.0
- 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 or later is required for editing.

The Geocortex Viewer for HTML5 2.5 uses the ArcGIS API for JavaScript 3.14.

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 1.4.

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 **9**, or
- **Install the Viewer Manually** on page **14**

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 **6** 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 (https://support.geocortex.com/).

To download the installation package for the HTML5 Viewer:

1. Open the Geocortex Support Center in your browser (https://support.geocortex.com).
2. If you do not have a Support account:
    a. Click the **Register** link on the Support Center's home page.
    b. Follow the instructions to set up your account.
    You will be sent an email acknowledging your request. When your account has been approved, you will be sent two emails—one with a password, and one with information about how to use the Support Center. You can sign in to the Support Center when you receive your password.
3. Sign in to the Support Center and click the **Downloads** tab.
4. In the **Geocortex Essentials** section, click the **Geocortex Viewer for HTML5** link.
   A list of the available versions of the HTML5 Viewer displays.
5. Click the `Geocortex Viewer for HTML5 2.5` link.
   Links to the HTML5 Viewer's documentation and installation package display.
6. Click the `Geocortex Viewer for HTML5 2.5.zip` link and save the file to your computer's hard drive.
7. Extract the files from the installation package.
   Familiarize yourself with the installation package by reading **Contents of the Installation Package** on page **8**.

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.2.5.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:**
  - **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.
  - **Administrator and Developer Guide.PDF:** A printable document on how to install, configure, and develop with 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 name of 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

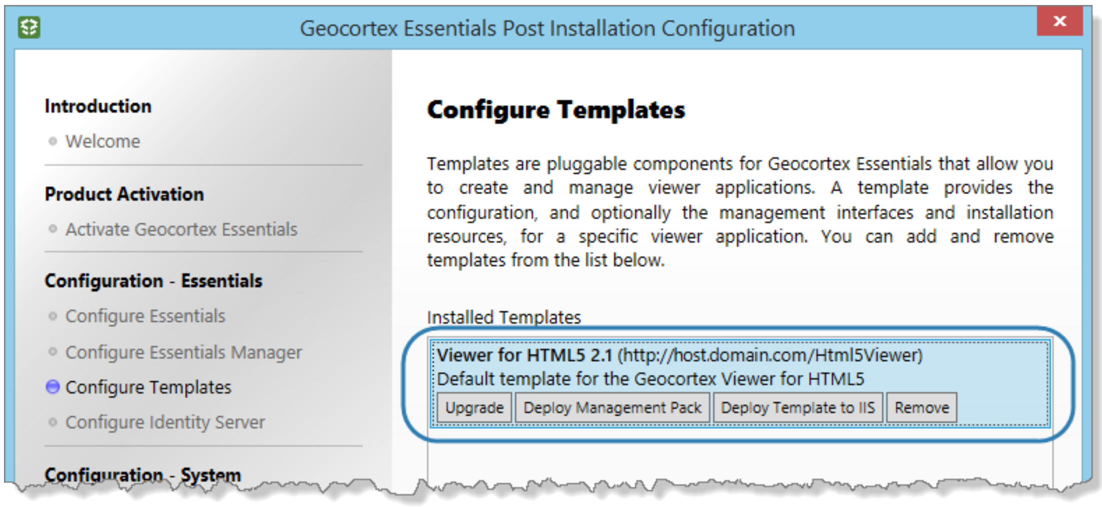
These instructions describe the steps to install the Geocortex Viewer for HTML5 using a viewer template.

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.


Geocortex Essentials Post Installation Configuration
Introduction
Welcome
Product Activation
Activate Geocortex Essentials
Configuration - Essentials
Configure Essentials
Configure Essentials Manager
Configure Templates
Configure Identity Server
Configuration - System
Upgrade
Deploy Management Pack
Deploy Template to IIS
Remove

Configure Templates

Templates are pluggable components for Geocortex Essentials that allow you to create and manage viewer applications. A template provides the configuration, and optionally the management interfaces and installation resources, for a specific viewer application. You can add and remove templates from the list below.

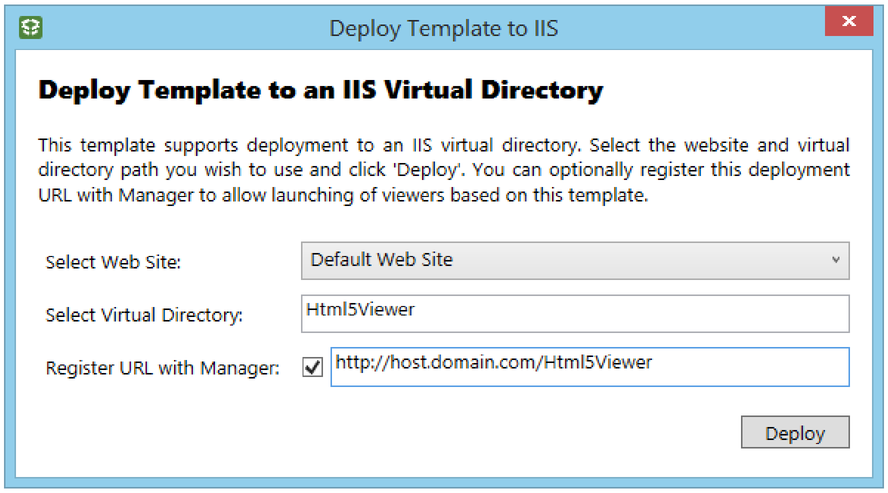
Installed Templates

Viewer for HTML5 2.1 (http://host.domain.com/Html5Viewer)
Default template for the Geocortex Viewer for HTML5

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 name of 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

Deploy Template to an IIS Virtual Directory

This template supports deployment to an IIS virtual directory. Select the website and virtual directory path you wish to use and click 'Deploy'. You can optionally register this deployment URL with Manager to allow launching of viewers based on this template.

Select Web Site:	Default Web Site
Select Virtual Directory:	Html5Viewer
Register URL with Manager:	✓ http://host.domain.com/Html5Viewer

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 Manager must have the same domain as Manager—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** check box is selected.

    If you clear the **Register URL with Manager** check box, the viewer launch links do not appear in Manager. You will still be able to use Manager to add and configure viewers.

12. If the URL beside the check box 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.

14. 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:

1. Click the **Windows Start** button, type **Essentials**, and click **Geocortex Essentials Manager**.
2. 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.

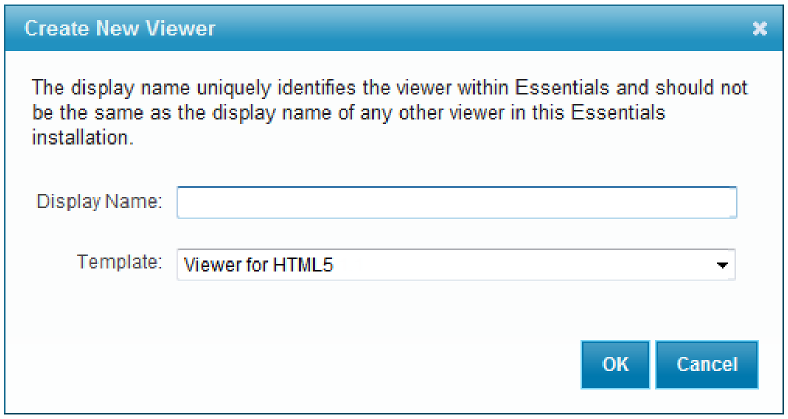
   If no sites exist, add a new site.

3. Edit the site that you want to add a viewer to.
4. In the sidebar, click **Viewers**.

   The Viewers page opens.

5. Click **Add Viewer**.

The Create New Viewer dialog box opens.



The display name uniquely identifies the viewer within Essentials and should not be the same as the display name of any other viewer in this Essentials installation.

Template: Viewer for HTML5

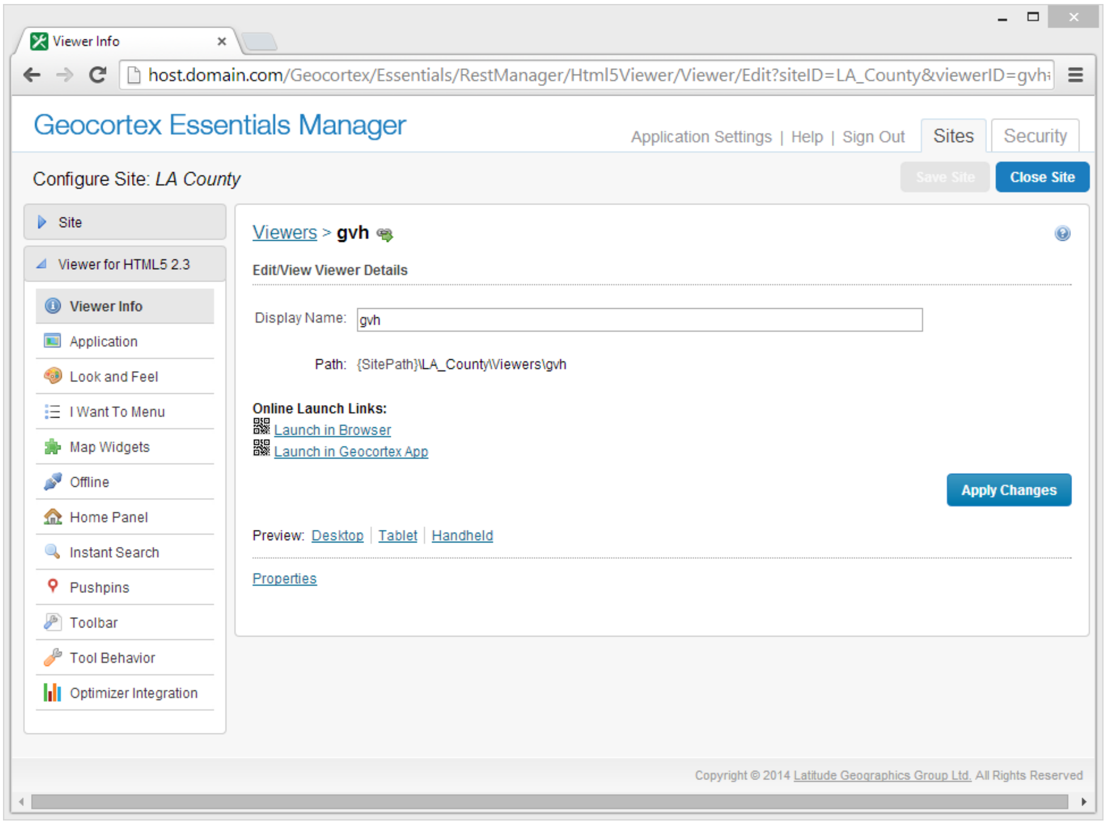
Create New Viewer dialog box

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

   Try keep the viewer's display name reasonably short. The viewer's folder is named after the viewer's ID, which is based on the viewer's display name. Windows folder names must be less than 248 characters and Windows path names must be less than 260 characters.

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

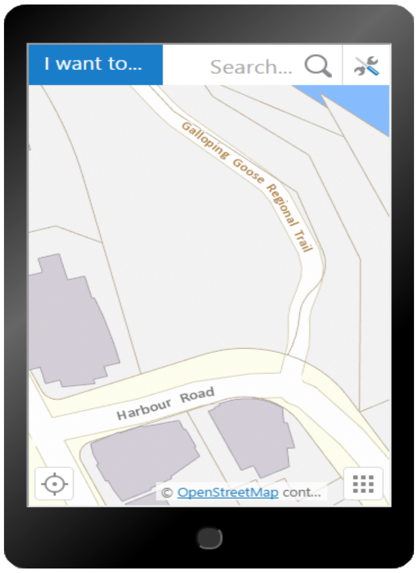
7. In the **Template** box, select **Viewer for HTML5 2.5** from the drop-down list.
8. Click **OK**.
   The Create New Viewer dialog box closes and Manager's Viewer Info page opens.



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

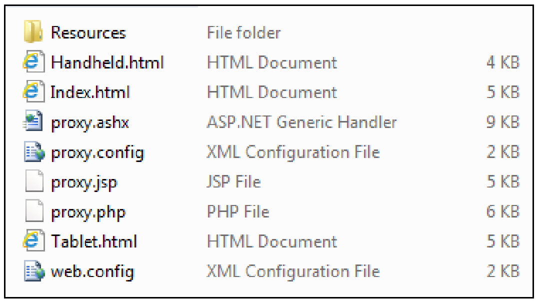
These instructions describe the steps to manually install the Geocortex Viewer for HTML5.

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

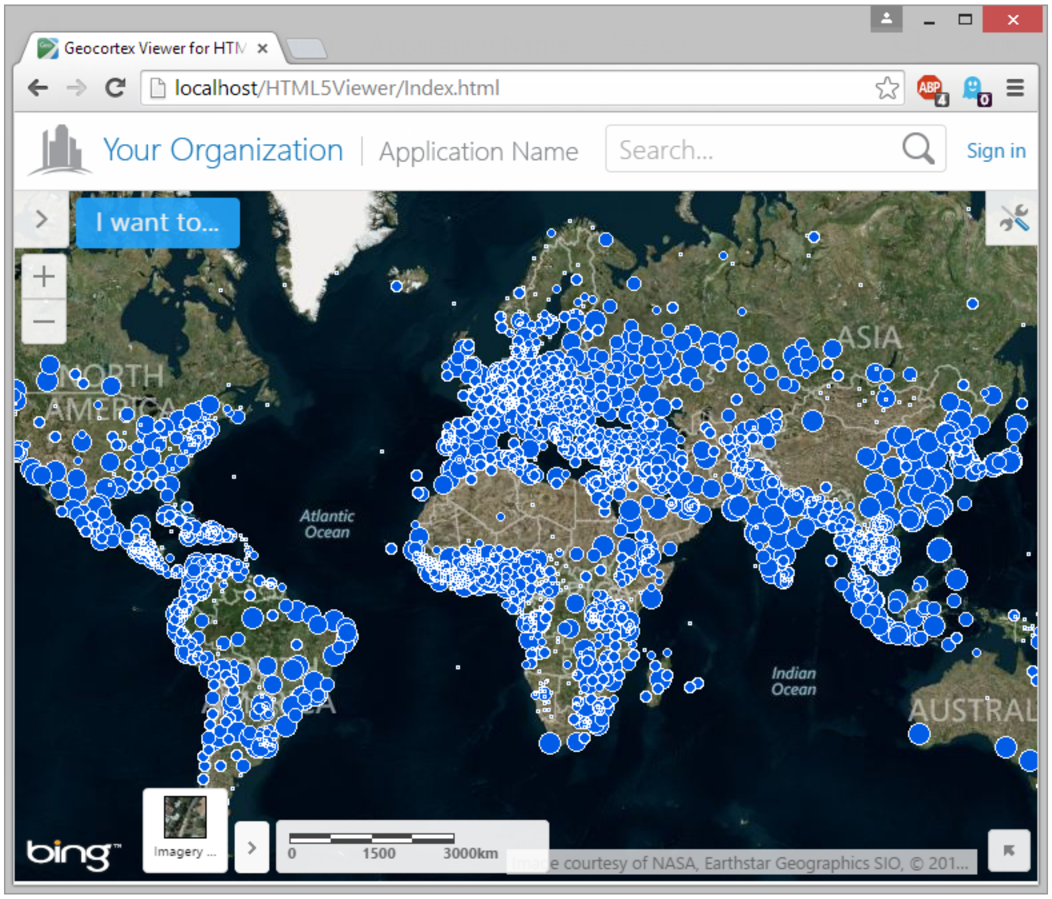
There is an HTML file in the deployment folder for each class of device that can be used to run the viewer:

- **`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

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 the Viewer Using Essentials** on page **17**.
  - Follow the instructions in **Upgrade Viewers Using Essentials** on page **18**.
- **Installed Manually:** If you installed the Viewer by dropping it directly into a web server:
  - Read **About Upgrading the Viewer Manually** on page **18**.
  - Follow the instructions in **Upgrade the Viewer Manually** on page **22**.
    This upgrade procedure retains the manual nature of your deployment.

8.1.1 About Upgrading the Viewer Using Essentials

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

1. Upgrade the viewer template.
2. Upgrade your configured HTML5 viewers to use the new template.

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 viewers—they cannot be launched or edited.

Upgrade the Viewer Template

The Post Installer has an Upgrade function that facilitates upgrading the viewer files that are deployed to the web server.

By default, the Post Installer's Upgrade function deploys the upgrade to the same location as the existing viewer. You can change the location where the upgrade is deployed if you want. Typically, you would deploy an upgrade to the same folder as the version that it is replacing, so the viewer's URLs remain the same. If you deploy 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. If, after upgrading, you decide that you want to go back to the old version of the viewer, you can use the backup to restore the old version. See **Restore a Viewer** on page **21** for instructions.

When a new version of the viewer template is released, some of the viewer files are different than in previous versions, as improvements are made to the viewer. If you have modified viewer files, such as HTML, CSS, JavaScript, or image files, you want to carry forward your modifications, but also benefit from any improvements that have been made. To do this, you may have to merge the two versions of a file together.

During the upgrade, the Upgrade function lists which of your customized files have changed in the new version of the viewer, so you can you manage how each file is upgraded. 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/.

After you have finished merging files, you specify which files to use. You have two choices:

- **Keep Existing:** Keep the installed version of the file and discard the new version.
- **Overwrite:** Use the new version of the file and discard the installed version.

Note that the list of files contains only those files that have changed both in the installed version and in the new version. If you modified a file, but it has not changed in the new release, then the installed version of the file is preserved, so you do not lose your changes.

Similarly, 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.

Viewer files that you have not customized are replaced when you upgrade. This means that you get any changes that have been made in the new versions of these files.

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 upgrade a manual installation of the HTML5 Viewer, you must back up the viewer's files, deploy the new viewer, and then manually copy over the changes from the backup files to the new files. You must also remove the current viewer from IIS.

8.2 Upgrade Viewers Using Essentials

If you installed the HTML5 Viewer manually, you cannot use this method to upgrade the viewer. See **Upgrade the Viewer Manually** on page **22**.

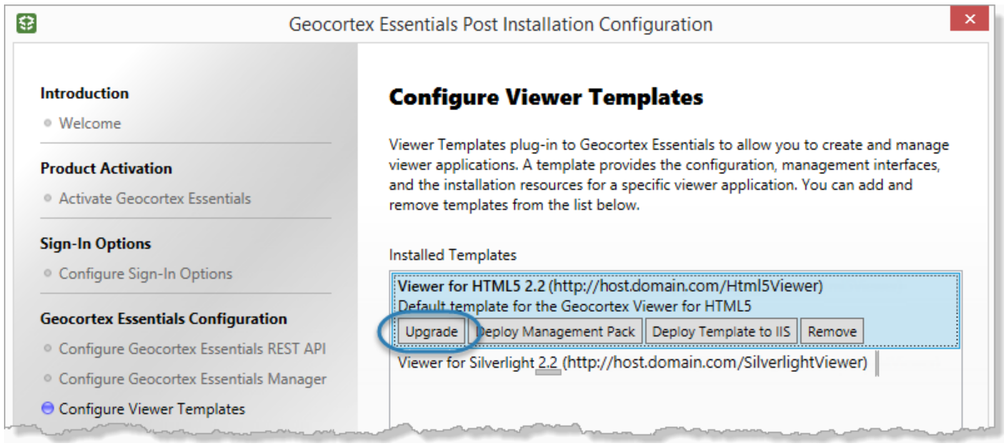
8.2.1 Step 1: Upgrade the Viewer Template

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 **7** 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.


Geocortex Essentials Post Installation Configuration
Introduction
Welcome
Product Activation
Activate Geocortex Essentials
Sign-In Options
Configure Sign-In Options
Geocortex Essentials Configuration
Configure Geocortex Essentials REST API
Configure Geocortex Essentials Manager
Configure Viewer Templates
Upgrade
Deploy Management Pack
Deploy Template to IIS
Remove

Configure Viewer Templates

Viewer Templates plug-in to Geocortex Essentials to allow you to create and manage viewer applications. A template provides the configuration, management interfaces, and the installation resources for a specific viewer application. You can add and remove templates from the list below.

Installed Templates

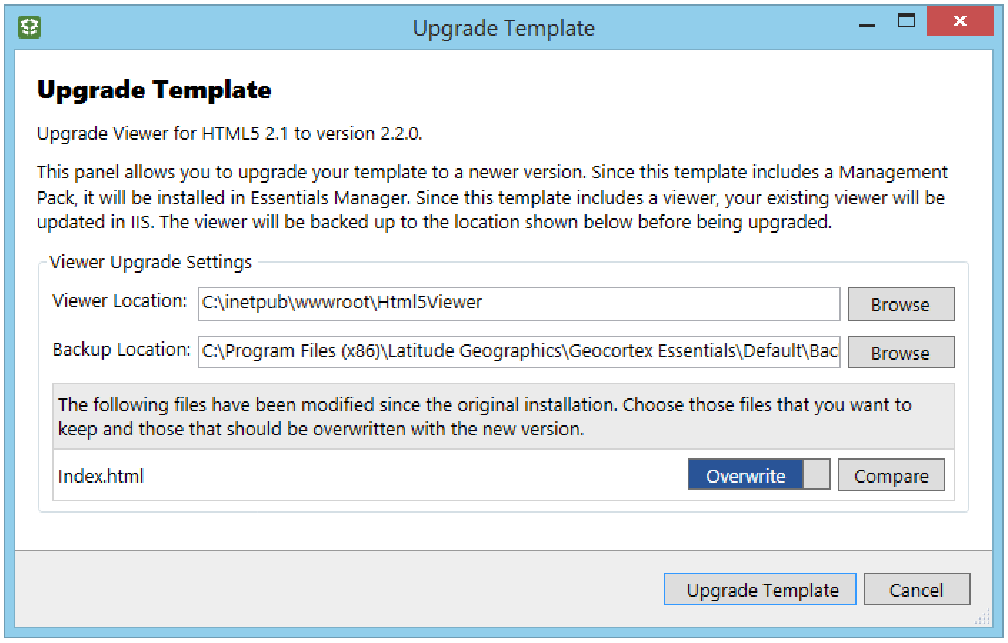
Viewer for HTML5 2.2 (http://host.domain.com/Html5Viewer)
Default template for the Geocortex Viewer for HTML5

Viewer for Silverlight 2.2 (http://host.domain.com/SilverlightViewer)

Location of the Upgrade button for upgrading a viewer template

3. Click **Upgrade**.
4. Browse to the location where you extracted the installation package.
5. Select the template file, `Geocortex.Essentials.HTML5Viewer.Template.[version].vte`, and then click **Open**.

   The Upgrade Template dialog box opens.



Upgrade Template

Upgrade Viewer for HTML5 2.1 to version 2.2.0.

This panel allows you to upgrade your template to a newer version. Since this template includes a Management Pack, it will be installed in Essentials Manager. Since this template includes a viewer, your existing viewer will be updated in IIS. The viewer will be backed up to the location shown below before being upgraded.

Viewer Upgrade Settings

Viewer Location: C:\inetpub\wwwroot\Html5Viewer

Backup Location: C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\Default\Bac

The following files have been modified since the original installation. Choose those files that you want to keep and those that should be overwritten with the new version.

Index.html

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. You can close the Post Installer.

8.2.2 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 **21**).

To upgrade all of your configured viewers at one time:

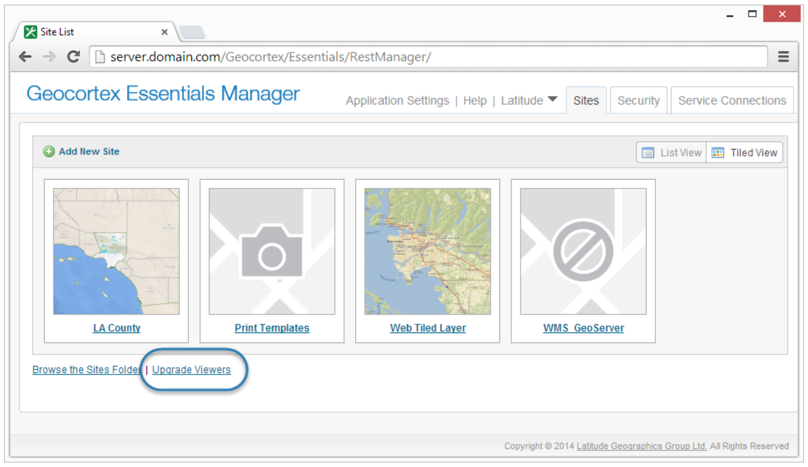
1. Do one of the following:

   #### If Manager is Closed

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

   #### If Manager is Open

   1. In Manager, click the **Sites** tab. If a site is open for editing, click **Save Site** if necessary, and then click **Close Site**.
   2. Click the **Upgrade Viewers** hyperlink on the **Sites** tab.


Site List
server.domain.com/Geocortex/Essentials/RestManager/
Geocortex Essentials Manager
Application Settings | Help | Latitude
Sites
Security
Service Connections
Add New Site
List View
Tiled View
LA County
Print Templates
Web Tiled Layer
WMS_GeoServer
Browse the Sites Folder | Upgrade Viewers
Copyright © 2014 Latitude Geographics Group Ltd. All Rights Reserved

Location of the Upgrade Viewers hyperlink

2. Make sure the HTML5 Viewer check box 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.

8.3 Restore a Viewer

If you upgraded the HTML5 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.

To restore a viewer, you must have a backup of the viewer, in other words, 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 was made automatically. By default, viewer backups created by the Upgrade function are in:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\Backups
```

To restore a previous version of the HTML5 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 `<GE_INSTALL>\Backups`.
2. Download the old template from the Geocortex Support Center (https://support.geocortex.com/).
   See **Download the Installation Package** on page **7** 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 name of the instance name, if Essentials is installed as a named instance. The default installation does not have an instance name.
4. Click **Configure Templates** in the side panel.
5. In the **Installed Templates** area, click the template that you want to roll back.
   The template's management buttons show.
6. Click **Remove**, and then click **Yes** to confirm.
7. Install the old template.
   See **Install the Viewer Framework Using a Viewer Template** on page **9** for instructions.
8. Copy the backup of the old viewer to the folder in the web server where the viewer was deployed.

8.4 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 **7**.
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 **16**.
6. Save the configuration files and test them by launching each of the layouts in a browser.
   See **Launch the Viewer** on page **15**.
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.

8.5 Upward Compatibility

Normally, your custom files are upwardly compatible—after you upgrade the viewer, the old files still work. Occasionally, improvements to the viewer break upward compatibility. In this case, you may have to modify or reconfigure a feature to make it work with the new version of the viewer.

8.5.1 Versions that Require Additional Steps

If you are upgrading any of the software versions listed below, you may have to take additional steps to complete the upgrade:

- **Upgrade the HTML5 Viewer from a Version that is Older than 2.4 - Viewer URLs** on page **23**
- **Upgrade the HTML5 Viewer to Version 2.5 or Newer - Offline** on page **26**

Upgrade the HTML5 Viewer from a Version that is Older than 2.4 - Viewer URLs

This section applies to you if you are upgrading from a pre-2.4 version of the HTML5 Viewer and you have not performed the steps for **Geocortex Viewer for HTML5 Security Update 2015-03-26**. If you performed the Security Update, or you installed the Geocortex Viewer for HTML5 for the first time in version 2.4 or newer, you can skip this section.

There are three main steps to perform Security Update 2015-03-26. You may have to perform one or more of these steps to complete a successful upgrade to Geocortex Viewer for HTML5 2.5. The steps are:

1. Apply the Security Update.
   This replaces the HTML5 Viewer's `Framework.js` file with a more secure version of the file.

   If you have not applied the Security Update, you do **not** need to apply it now. The new version of `Framework.js` is built into Geocortex Viewer for HTML5 2.4 and newer. When you upgrade the Viewer to 2.4 or newer, the new version of the file replaces the old version.

2. If you use the `configBase` parameter in 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 Geocortex Viewer for HTML5 2.4 or newer. Follow the instructions in **Load Files from the Same Domain** on page **24**.

3. If you use CORS to allow cross-origin resource sharing between Essentials and your HTML5 viewers, configure the viewers' trusted domains.

   If you have not configured trusted domains for your viewers, the viewers may not work in Geocortex Viewer for HTML5 2.4 or newer. Follow the instructions in **Load Files from a Different Domain** on page **25**.

For more information about Geocortex Viewer for HTML5 Security Update 2015-03-26, refer to the Geocortex Knowledge Base article, CORS Security Update.

Load Files from the Same Domain

Geocortex HTML5 2.4 and newer viewers will not load a configuration file that is hosted in the same domain as the viewer if the domains do not match exactly. For example, the following URL will fail to launch a version 2.4 or newer HTML5 viewer because the viewer's domain, `myserver`, does not exactly match the configuration's domain, `myserver.mydomain.com`, even though they map to the same machine:

```
http://myserver/Html5Viewer/Index.html?configBase=http://myserver.mydomain.com/Geocortex/…
```

Invalid viewer URL with non-matching domains

If your environment is configured to produce URLs with domains that do not match, you may have to update the configuration to make the URLs work with version 2.4 and newer. Specifically, if CORS is set up on your server or your server is behind a proxy, you must update the configuration.

We recommend that you use fully qualified domains in your URLs, even if you are not using CORS or a proxy.

The procedure below updates the viewer launch links that appear in Manager.

In addition to updating the viewer launch links in Manager, you must update the launch links that you provide to end users.

To update the launch links in Manager to use fully qualified domains:

1. Update the HTML5 Viewer framework's base launch link:
    a. Run an XML editor or text editor as an administrator.
    b. Open the `RestManagerAppSettings.xml` file in the editor.
       By default, the file is located here:
       ```
       C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\
       [instance]\REST Elements\Manager\App_Data\RestManagerAppSettings.xml
       ```
    c. In the HTML5 Viewer's `ViewerFramework` element, update the `Url` attribute to use a fully qualified domain, for example:
       ```
       <ViewerFramework ProductID="..." TemplateID="Html5Viewer_2_5"
       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.
       By default, the file is located here:
       ```
       C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\
       [instance]\REST Elements\Manager\web.config
       ```
    b. In 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 a Different Domain

If you use CORS to allow cross-origin resource sharing between Essentials and HTML5 viewers, starting in version 2.4 of the HTML5 Viewer, you must configure trusted domains for your viewers. The trusted domains list is configured in a viewer's host page. You must configure the trusted domains for each viewer.

To specify trusted domains for an HTML5 viewer:

1. Run an HTML editor or text editor as an administrator.
2. Open the viewer's host file in the editor.
   The default host page for HTML5 viewers is `Index.html`. By default, the host file is located here:
   ```
   C:\inetpub\wwwroot\Html5Viewer\Index.html
   ```
3. Find where the viewer is created:
   ```
   var viewer = new geocortex.essentialsHtmlViewer.ViewerApplication
   (viewerConfig.viewerConfigUri, null, viewerId);
   ```

4. Immediately before the section where the viewer is created, specify your trusted domain(s) using the `geocortex._configDomains` object, for example:

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

5. Save the file.

6. Repeat these steps for each host file in each HTML5 viewer.

Upgrade the HTML5 Viewer to Version 2.5 or Newer - Offline

As of HTML5 Viewer 2.5, the 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 HTML5 Viewer is hosted. For example, `C:\inetpub\wwwroot\`**`Html5Viewer`**.
- The virtual directory of the HTML5 Viewer on the Essentials server. For example, `C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\`**`Default`**`\REST Elements\Sites\`**`MySite`**`\Viewers\`**`MyViewer`**`\VirtualDirectory`.

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
```

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). Note that Internet Explorer 8 and Internet Explorer 9 do not fully support CORS, so you must set up a proxy page if you want to support these browsers.
  To enable cross-origin access when the HTML5 Viewer is deployed to different domain than Essentials, use CORS.

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 **28**.

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

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. Right-click the Viewer in the **Connections** panel and select **Switch to Content View**.
   The center panel shows the content.
3. In the center panel, right-click `proxy.ashx` and select **Switch to Features View**.
   The center panel shows the Features View.
4. In the **IIS** area, double-click the **Authentication** icon .
   The center panel shows the authentication settings.
5. Right-click **Anonymous Authentication** and select **Disable**.
6. 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.

7. 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)

An Essentials system can have multiple servers, with applications and other resources on different domains. Web servers and web browsers restrict cross-domain requests for security reasons. If your system is hosted on multiple domains, you must set up Cross-Origin Resource Sharing (CORS) so that browsers running the HTML5 Viewer do not block requests.

CORS was introduced in HTML5 as a way to allow legitimate interactions between origins. The Wikipedia article Cross-origin resource sharing provides an introduction to CORS.

For information about setting up cross-domain access for Silverlight viewers, refer to the *Geocortex Viewer for Silverlight Administrator and Developer Guide*.

Two common scenarios that use CORS are:

- Essentials and the HTML5 Viewer are deployed to different domains. By default, the HTML5 Viewer will not load configuration from a different domain, and Essentials will not allow cross-domain requests.

  For example, you may want to embed the HTML5 QuickStart viewer into an existing site that is on a different domain. (The QuickStart viewer is included in the HTML5 Viewer's installation package.)

- The viewer accesses resources, such as workflows or a geometry service, that are on a different domain than the Viewer. In this case, you can use a proxy page to set up cross-origin access instead of using CORS.

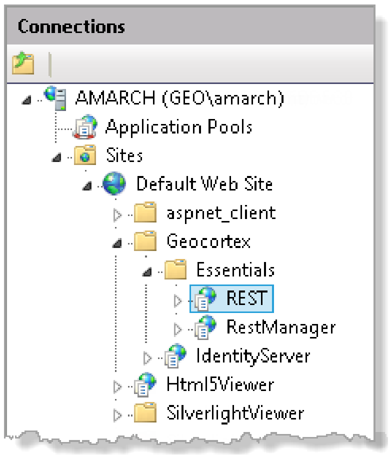
We recommend deploying Essentials and the HTML5 Viewer to the same domain.

To set up Cross-Origin Resource Sharing between Essentials and HTML5 viewers:

These instructions assume that you are running Geocortex Viewer for HTML5 version 2.4 or newer.

Step 1: Configure CORS in Essentials

1. In IIS Manager, expand the folders in the **Connections** panel and select the **REST** application.

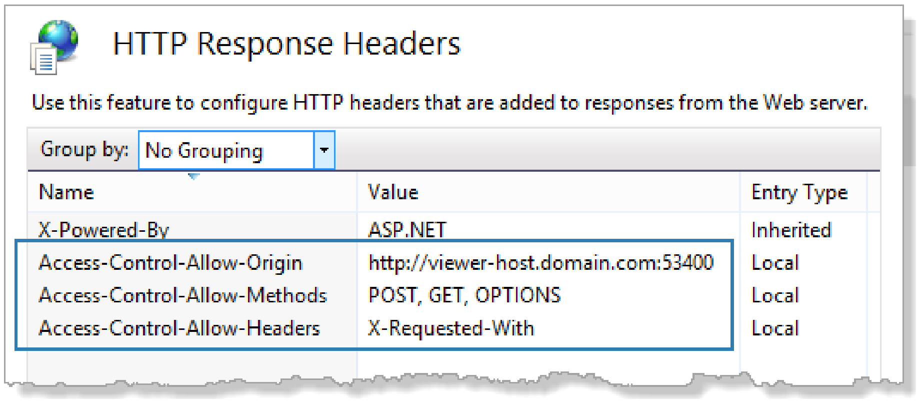


Default location of the REST application

2. In the center panel's **IIS** section, double-click **HTTP Response Headers**.
3. Add the following response headers:
    - **Access-Control-Allow-Origin:** Origin where the HTML5 Viewer is hosted
    - **Access-Control-Allow-Methods:** POST, GET, OPTIONS
    - **Access-Control-Allow-Headers:** X-Requested-With

For security, do not use the Allow All Origins wildcard (*) in the Access-Control-Allow-Origin response header, and do not configure addition methods or headers.

To add a response header, click **Add** in the **Actions** panel. These headers are sufficient for most installations.


HTTP Response Headers

Use this feature to configure HTTP headers that are added to responses from the Web server.

Group by: No Grouping

Name	Value	Entry Type
X-Powered-By	ASP.NET	Inherited
Access-Control-Allow-Origin	http://viewer-host.domain.com:53400	Local
Access-Control-Allow-Methods	POST, GET, OPTIONS	Local
Access-Control-Allow-Headers	X-Requested-With	Local

HTTP response headers to configure for CORS

4. Click IIS Manager's **Back** button to return to the **REST** application's home page.
5. In the center panel's **IIS** section, double-click **Handler Mappings**.
6. Scroll down to see if **OPTIONSVerbHandler** appears in the **Name** column.
   If OPTIONSVerbHandler is there, skip this step.

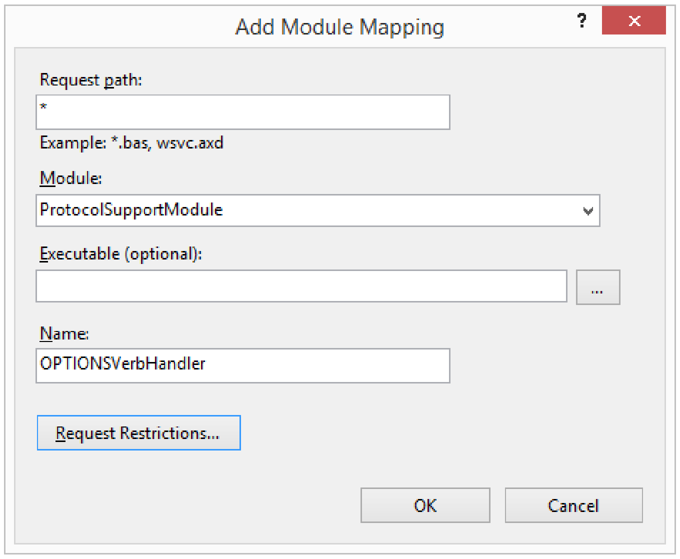
If OPTIONSVerbHandler does not appear in mappings, configure it now:

a. In the **Actions** panel, click **Add Module Mapping**.

b. In the **Request Path** box, type an asterisk, *****.

c. Select **ProtocolSupportModule** from the **Module** drop-down list.

d. In the **Name** box, type **OPTIONSVerbHandler**.



Request path: *
Module: ProtocolSupportModule
Name: OPTIONSVerbHandler

Handler mapping for the HTTP OPTIONS method

e. Click **Request Restrictions**.

f. On the **Mapping** tab, clear the **Invoke handler only if request is mapped to** check box.

g. On the **Verbs** tab, select **One of the following verbs** and type **OPTIONS** in the box.

h. Click **OK**.

i. Click **OK**.

7. Close IIS Manager.

Step 2: Configure trusted origins in your HTML5 viewers

1. Run an HTML editor or text editor as an administrator.

2. Open the viewer's host file in the editor.
   The default host page for HTML5 viewers is `Index.html`. By default, the host file is located here:
   `C:\inetpub\wwwroot\Html5Viewer\Index.html`

3. Find where the viewer is created:

```
var viewer = new geocortex.essentialsHtmlViewer.ViewerApplication
(viewerConfig.viewerConfigUri, null, viewerId);
```

4. Immediately before the section where the viewer is created, 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 file.
6. Close the file.
7. Repeat these steps for each host file and each HTML5 viewer.

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 **263**

UI Composition – Regions and Views on page **266**

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:

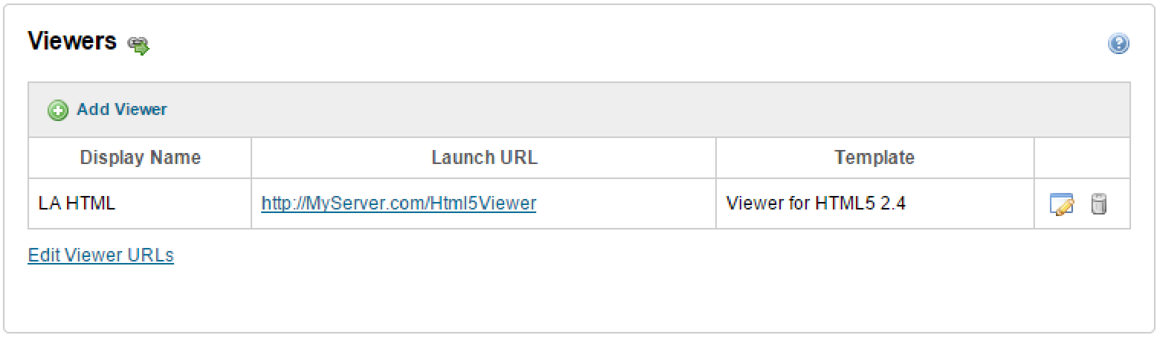
```
http://<server.mydomain.com>/<iis-virtual-directory>
```

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 Viewer section:



Display Name	Launch URL	Template	
LA HTML	http://MyServer.com/Html5Viewer	Viewer for HTML5 2.4	

When you click on 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 box 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_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 HTML5 Viewer launch URL has two URL 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.

Example

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

```
http://server.domain.com/vwr/Handheld.html?viewerConfigUri=http://server.doma
in.com/Geocortex/Essentials/REST/sites/LA_County/viewers/LA_
HTML/VirtualDirectory/Resources/Config/Default/Handheld.json.js&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 URL Parameters Reference

NOTE 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 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://server.domain.com/vwr/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://server.domain.com/vwr/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://server.domain.com/vwr/Index.html?`
`center=255421.6566,6250846.4361,3857&scale=9000`

Module: Map Module

`configBase`

Specifies the URI of the folder where the configuration files are stored. The viewer automatically detects the device type and loads the appropriate configuration file from the specified folder. We recommend using 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_HTML` viewer and loads the appropriate configuration file for the device type. This is how you would use `configBase` in a production viewer.

`http://server.domain.com/vwr/Index.html?`**`configBase=http://server.domain.com/`**
`Geocortex/Essentials/REST/sites/LA_County/Viewers/LA_`
`HTML/VirtualDirectory/Resources/Config/Default`

Example 2: This example launches the `LA_County` site's `LA_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.

`http://server.domain.com/vwr/Handheld.html?`

URL Parameters that can be Used in HTML5 Viewer URLs

```
configBase=http://server.domain.com/Geocortex/Essentials/REST/sites/LA_
County/Viewers/LA_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://server.domain.com/vwr/Index.html?debug=true
```

Module: Log Module

`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://server.domain.com/vwr/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://server.domain.com/vwr/Index.html?extent=35.2468,-80.8718,35.2139,-
80.8210,4326
```

Module: Map Module

`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://server.domain.com/vwr/Index.html?layerTheme=0
```

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

```
http://server.domain.com/vwr/Index.html?layerTheme=Imagery
```

Module: LayerThemes Module

URL Parameters that can be Used in HTML5 Viewer 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 **258** for instructions.

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://server.domain.com/vwr/Index.html?locale=fr-CA
```

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

`run` or `runWorkflow`

Runs the specified workflow(s), mapping URL parameters to workflow input arguments. The `run` and `runWorkflow` URL parameters are the same—you can use either.

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

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://server.domain.com/vwr/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://server.domain.com/vwr/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.

URL Parameters that can be Used in HTML5 Viewer URLs

`http://server.domain.com/vwr/Index.html?`**`run=FindParcel&pid=02503116`**

Module: Workflow Module

`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://server.domain.com/vwr/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.

```
http://server.domain.com/vwr/Index.html?
center=255421.6566,6250846.4361,3857&scale=9000
```

Module: Map Module

`viewer`

Specifies the ID of the viewer to launch. The viewer ID and location of the viewer's configuration files must be specified in the `Index.html` file. The configuration files must be on the same domain as the viewer.

Syntax: `viewer=viewerId`

Example: In this example, the viewer with ID "la_html" launches, provided the viewer ID is mapped to the location of the viewer's configuration files in `Index.html`:

`http://server.domain.com/vwr/Index.html?`**`viewer=la_html`**

Module: None

`viewerConfigUri`

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_HTML viewer using the Handheld interface:

```
http://server.domain.com/vwr/Index.html?viewerConfigUri=http://server.domain.com/
Geocortex/Essentials/REST/sites/LA_County/viewers/LA_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.

URL Parameters that can be Used in HTML5 Viewer URLs
Module: None

12.3 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 **33**. 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`
`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** tab, and then click **Providers** in the side panel.
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.htm
l?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 **9**.
- **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 **95**.

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 **49** for instructions.

For information about configuring application-wide properties, see **Application-Wide Settings** on page **43** and **Configure a Viewer in Manager** on page **49**.

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 **292**.

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 **43** 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 **45** for information.
- `libraries`**:** An array containing the libraries that make up the application. See **Libraries and the Default Library ID** on page **45** for information.
- `modules`**:** An array of modules in the application. See **Modules** on page **45** for information.
  - `views`**:** An array containing the module's views. See **Views** on page **45** for information.
  - `viewModels`**:** An array containing the module's view models. See **View Models** on page **46** 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 **52**.

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 **26**.

- **`allowUnsafeContent:`** When this property is set to `true`, 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, set `allowUnsafeContent` to `false`. The default value is `false`.

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

- **`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.

- **`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, Manager 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 using the `geometryServiceUrl` property.

  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`).

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 **96** and the relevant subsection of **Configuration Settings by Module** on page **95**. For example, for the Banner Module, see **Common Settings for Modules** on page **96** and **Banner Module** on page **101**.

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 **96**.

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 **97**.

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 **95** 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 **43**.

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 **95** 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**.

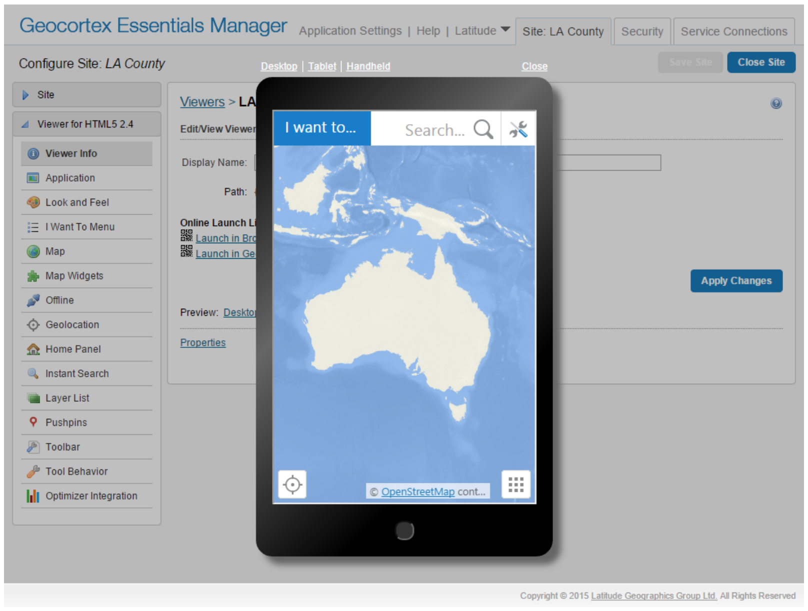
   The tabs are combined into one tab, and the configurations for this page's settings are made the same.

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 two language files, one for each library. 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.

When you configure user interface text in 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 **258**.

  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 **258**.

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 **9** 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.

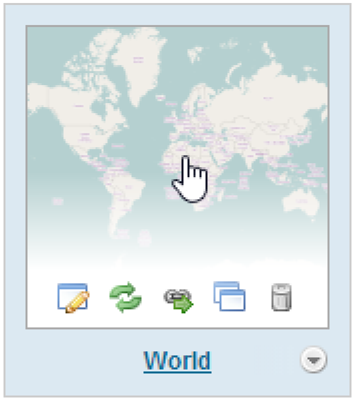
   Try keep the viewer's display name reasonably short. The viewer's folder is named after the viewer's ID, which is based on the viewer's display name. Windows folder names must be less than 248 characters and Windows path names must be less than 260 characters.
4. In the **Template** drop-down list, select 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 **9** 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 Edit a Viewer

To edit a viewer:

1. Launch Manager. Follow the instructions that apply to you:
   - **Windows Server 2012 or Windows 8, and Newer Versions:**
     On the **Start** screen, type **Geocortex**, and then click **Geocortex Essentials Manager**.
   - **Windows Server 2008 or Windows 7, and Older Versions:**
     In the **Start** menu, select **All Programs** | **Latitude Geographics** | **Geocortex Essentials [Version] [Instance]** | **Geocortex Essentials Manager**.
     [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.

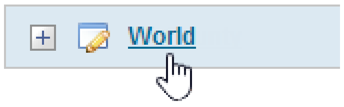
2. Edit the site whose viewer you want to configure:
   a. Click the **Sites** tab.
   b. If a different site is open for editing, click **Save Site**, and then click **Close Site**.
   c. Follow the instructions for the view of the Site List that you are using:
      - **Tiled View:**



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 click the **Edit** icon.

      - **List View:**



World site in List View

- Click the site's name, or
- Click 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.

14.3 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:

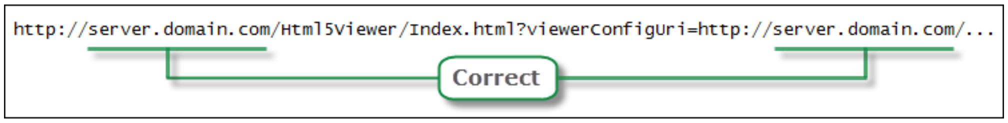
- **Display Name:** Used to refer to the viewer in Manager and in the REST API Sites Directory. The display name is not visible to the end user, however, the viewer's ID can be—it can appear in the viewer's URL. The ID is based on the display name at the time the viewer is created. Changing the display name after the viewer has been created does not change the ID.

  Try keep the viewer's display name reasonably short. The viewer's folder is named after the viewer's ID, which is based on the viewer's display name. Windows folder names must be less than 248 characters and Windows path names must be less than 260 characters.

- **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.
- **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 **40**
  - **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 **9** 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.


http://server.domain.com/Html5Viewer/Index.html?viewerConfigUri=http://server.domain.com/...
Correct

Recommended form for an HTML5 viewer URL

- **Preview:** (available on all HTML5 Viewer configuration pages in Manager) 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. The previews are live—you can do everything in the previews that you can do if you launch the viewer in a browser.
  If the previews are blank, see the note under Online Launch Links, above.
- **Properties:** (available on all HTML5 Viewer configuration pages in Manager) 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**.

14.4 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 **43**.

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 **47**.

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 **26**.
- **Allow Unsafe Content:** If this check box 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 check box. 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:

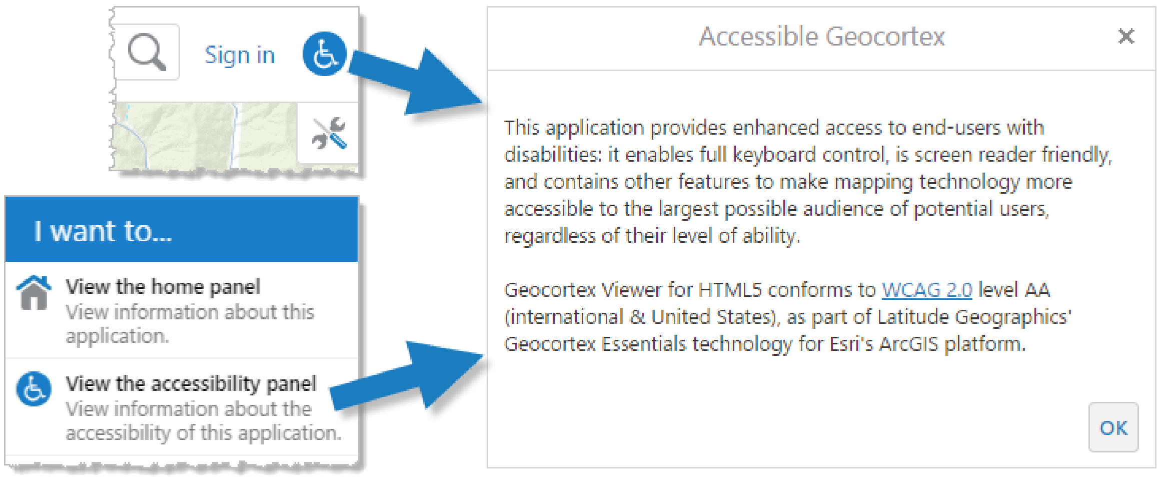
1. Click **Apply Changes**.
2. To save your changes, click **Save Site**.

14.5 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 clicks the **Accessibility Button** at the top-right of the viewer. In the Handheld interface, the user clicks the **View the accessibility panel** option in the **I Want To Menu**.

You can configure the viewer to show or hide the Accessibility Button, as well as 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.


Sign in
I want to...
View the home panel
View information about this application.
View the accessibility panel
View information about the accessibility of this application.
Accessible Geocortex
OK

This application provides enhanced access to end-users with disabilities: it enables full keyboard control, is screen reader friendly, and contains other features to make mapping technology more accessible to the largest possible audience of potential users, regardless of their level of ability.

Geocortex Viewer for HTML5 conforms to WCAG 2.0 level AA (international & United States), as part of Latitude Geographics' Geocortex Essentials technology for Esri's ArcGIS platform.

To open the Accessibility Panel, click the Accessibility Button in the Desktop or Tablet interfaces (top-left), or the I Want To Menu option in the Handheld interface (bottom-left)

The Accessibility Panel is implemented by the Accessibility Module. See **Accessibility Module** on page **97** for information.

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

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **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 **47**.

Settings

The Accessibility page has the following settings:

- **Include Accessibility Button:** (Desktop and Tablet interfaces only) When this checkbox is selected, the Accessibility Button is available in the viewer. If you do not want the viewer to have an Accessibility Button, clear the checkbox. By default, the viewer has an Accessibility Button.

- **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 **48**.

  The default title is **@language-accessibility-map-title**, which by default reads **Accessible Geocortex**.

- **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.

  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.

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 **253**

Accessibility Module on page **97**

About User Interface Text on page **48**

14.6 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
- Data Frame
- Results List
- Map Tips
- Feature Details

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 **47**.

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 **48** 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.
- **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.

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 **226**.

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**.

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 **48** for instructions.
3. To save your changes, click **Save Site**.

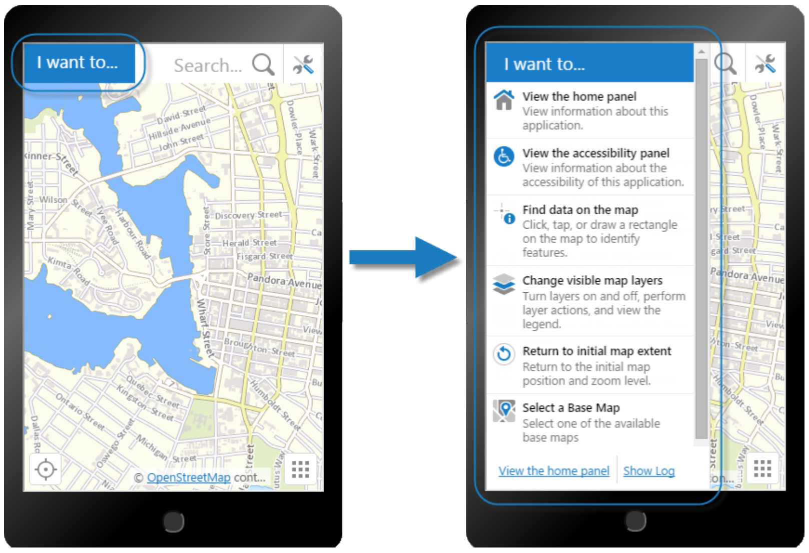
See Also...

About User Interface Text on page **48**

Configure the Home Panel on page **71**

14.7 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...
Search...
© OpenStreetMap cont...
I want to...
View the home panel
View information about this application.
View the accessibility panel
View information about the accessibility of this application.
Find data on the map
Click, tap, or draw a rectangle on the map to identify features.
Change visible map layers
Turn layers on and off, perform layer actions, and view the legend.
Return to initial map extent
Return to the initial map position and zoom level.
Select a Base Map
Select one of the available base maps
View the home panel
Show Log

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.

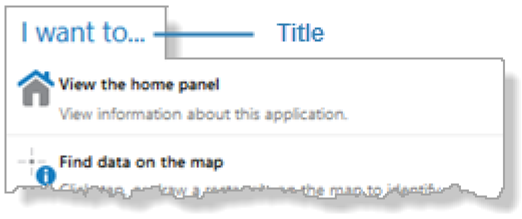
For more information, see **About Configuring Multiple Interfaces** on page **47**.

Settings

The I Want To Menu page has the following settings:

- **Show Menu:** If this check box is selected, the I Want To menu appears in the viewer. If the check box is cleared, the viewer does not have an I Want To menu.
- **Title:** 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 **48** for more information on using text keys.

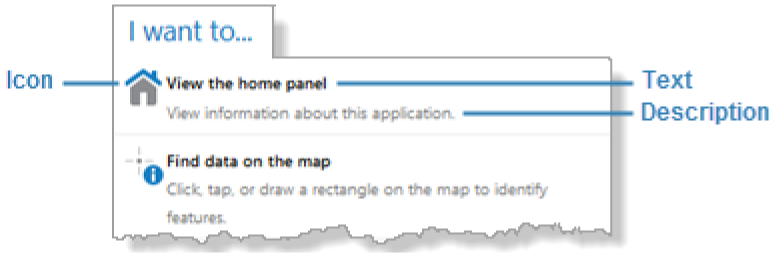


Title for the I Want To menu

To add an item to the I Want To menu:

1. Click **Add Menu Item**.
   The Add Menu Item dialog box opens.

2. **Appearance:** Configure the menu item's appearance:


I want to...
Icon
View the home panel
View information about this application.
Text
Description
Find data on the map
Click, tap, or draw a rectangle on the map to identify features.

Parts of I Want To menu items

- **Text:** The menu item's text.
- **Description:** (optional) A short description of what the menu item does.
- **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.
  To browse to the image file:
  a. Click **Browse**.
     The Select File dialog box opens.
  b. Expand the **Resources** | **Images** branch of the **VirtualDirectory** tree in the side panel.
  c. Select the **Images** folder.
  d. Click **Upload**.
     The Upload dialog box opens.
  e. Click **Select**.
  f. Browse to and select the image file you want to use for the menu item.
  g. Click **Open**.
  h. Click **Upload**.
     The file appears in the Filename list.
  i. With the file selected, click **OK**.

3. **Command:** Configure the command that the menu item runs:
   - **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.

   For more information on the HTML5 Viewer's commands and their parameters, see the Geocortex SDK for HTML5 API Reference.
4. Click **OK**.

To edit an I Want To menu item's appearance or command:

1. Click the **Edit** icon next to the item you want to modify.

   The Edit Menu Item dialog box opens.

2. Edit the details as desired.
3. Click **OK**.

To remove an item from the I Want To menu:

⚠ Removing menu items is permanent. If you remove a menu item, you cannot get it back after you have applied your changes.

1. Click the **Remove** icon beside the item you want to remove.

   You are prompted to confirm.

2. Click **OK**.

   The item is removed from the list of menu items.

To change the order of items in the I Want To menu:

The menu items are listed in Manager in the same order that they appear in the viewer.

1. Click and hold the icon for the menu item whose position in the menu you want to change.
2. Drag the item to its new position.
3. Repeat these steps until the menu is ordered the way you want it.

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 **48** for instructions.

3. To save your changes, click **Save Site**.

See Also...

About User Interface Text on page **48**

Geocortex SDK for HTML5 API Reference on page **269**

14.8 Configure the Map

The 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 **47**.

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.

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 **48** for instructions.
3. To save your changes, click **Save Site**.

14.9 Configure Map Widgets

The Map Widgets page in Manager has settings for:

- Bookmarks
- Scale Bar
- Overview Map
- Map Coordinates

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 **47**.

Settings

The Map Widgets page has the following settings:

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 **79**. 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.

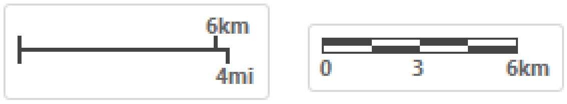
  By default, the Show Bookmarks Button checkbox is cleared.

  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**.

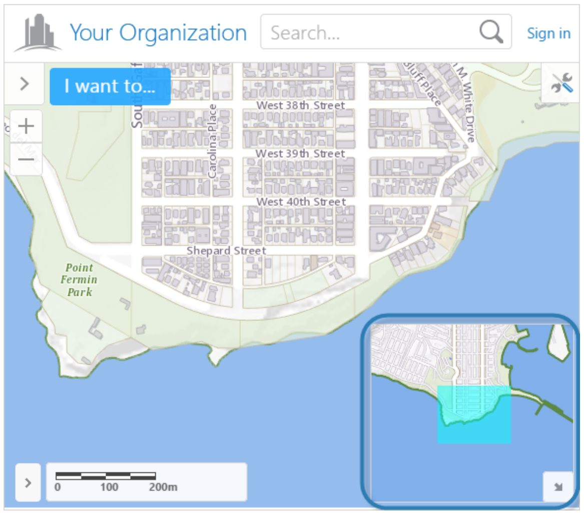


Dual-unit Line scale bar (left) and metric Ruler scale bar (right)

- **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.

Overview Map:

The overview map appears at the bottom right of the map.

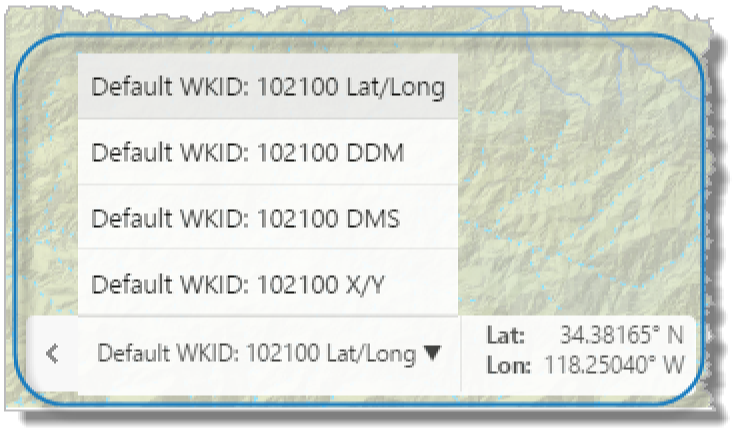


Example of an overview map

- **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.
- **Display Basemap's Coordinate Systems:** To include the base map's coordinate system in the coordinates widget, select this checkbox. This creates four options that the user can select from—display lat/lon, DDM, DMS, or X/Y coordinates. The Display Basemap's Coordinate Systems checkbox is selected by default.

  If you do not want to the coordinates widget to show the coordinates in the base map's coordinate system, clear the Display Basemap's Coordinate Systems checkbox and configure at least one Alternate Coordinate System.
- **Coordinate Decimal Precision:** The number of decimal places to use for coordinates. The default is **5**.
- **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)

The Map Coordinates widget is only available for the Desktop interface.

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 **48** for instructions.
3. To save your changes, click **Save Site**.

14.10 Configure a Viewer for Offline Use

Through the Geocortex Mobile App Framework, the HTML5 viewer supports access to feature layers and base maps 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.

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.

Before you configure settings on the Offline 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 Offline 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 **47**.

Settings

The Offline page has the following settings:

- **URL:** If you have deployed the viewer template to IIS multiple times, you can select which deployment you want the launch link and QR code to use. This option only appears if the template is deployed more than once.
- **Advertise viewer in Geocortex Mobile App Framework:** Whether this viewer should be offered to be added in the Geocortex Mobile App Framework.
- **Automatically download in Geocortex Mobile App Framework:** Whether this viewer should be automatically downloaded in the Geocortex Mobile App Framework.
- **Select Base Maps:** The HTML5 Viewer supports the offline use of base maps when the Viewer is running in the Geocortex Mobile App Framework. If you are going to run the Viewer in a browser, or if your site does not have any base maps, this setting does not apply to you.

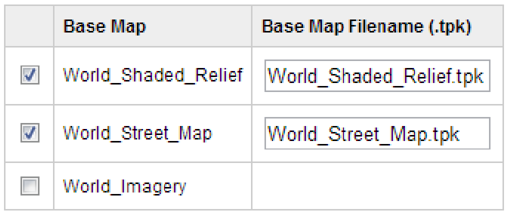
To configure an HTML5 viewer for offline use:

1. If you have multiple deployments of the HTML5 viewer in IIS, select the deployment that you want the launch link and QR code to use.
2. To offer this viewer to be added in the Geocortex Mobile App Framework, select the **Advertise viewer in Geocortex Mobile App Framework** checkbox.
3. To automatically download this viewer in the Geocortex Mobile App Framework, select the **Automatically**

download in Geocortex Mobile App Framework checkbox.

4. In the **Base Map** table, select the checkbox beside each base map that you want to be available offline.
5. For each base map you select, type the TPK's file name in the **Base Map Filename (.tpk)** column.

   You must enter the full file name, including the **.tpk** file extension. For example, **imagery.tpk**.

	Base Map	Base Map Filename (.tpk)
✓	World_Shaded_Relief	World_Shaded_Relief.tpk
✓	World_Street_Map	World_Street_Map.tpk
☐	World_Imagery	

   Two base maps configured for offline use

6. Follow the instructions in the *Geocortex Mobile App Framework Administrator Guide* to distribute Tile Packages (TPK files) to end users to load on their devices.
7. Click **Apply Changes**.
   The end user can now set up offline use of the viewer.

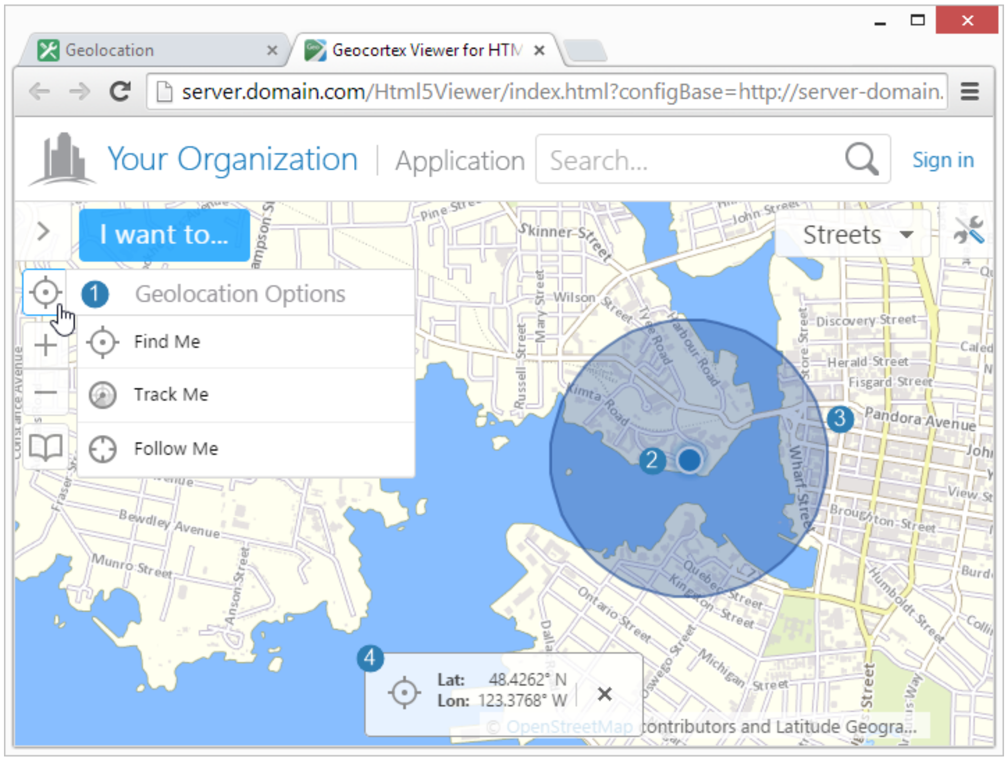
Before you leave the Offline page:

1. Click **Save Site**.

14.11 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)

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 **52**.

NOTE Internet Explorer 8 does not support geolocation.

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 **144**.

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 **47**.

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)

    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.
- **Coordinate Decimal Precision:** The number of decimal digits to show in the coordinates.

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 **48** for instructions.
3. To save your changes, click **Save Site**.

See also...

Geolocate Module on page **144**

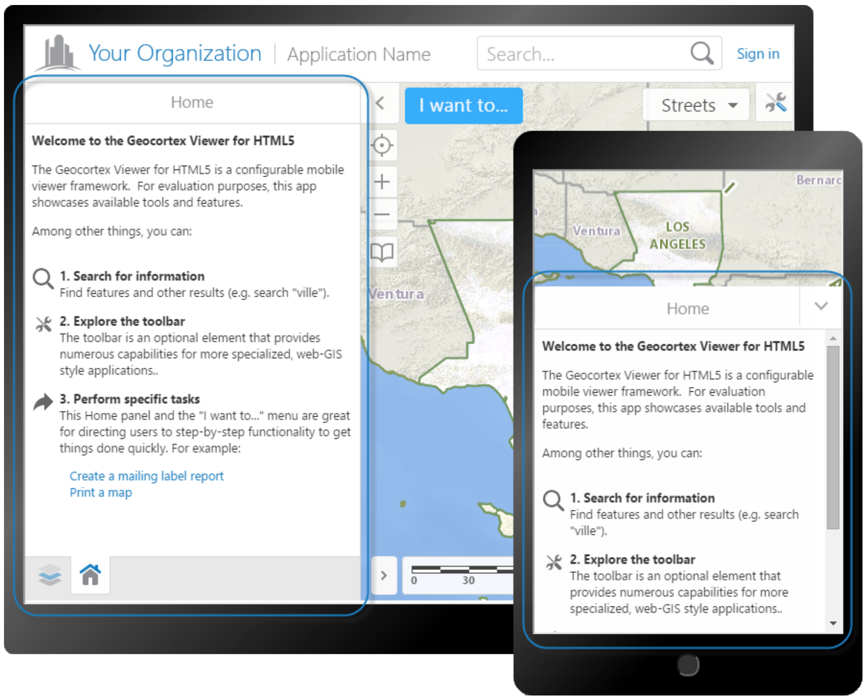
14.12 Configure the Home Panel

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.

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.



Welcome to the Geocortex Viewer for HTML5

The Geocortex Viewer for HTML5 is a configurable mobile viewer framework. For evaluation purposes, this app showcases available tools and features.

Among other things, you can:

1. Search for information
Find features and other results (e.g. search "ville").

2. Explore the toolbar
The toolbar is an optional element that provides numerous capabilities for more specialized, web-GIS style applications..

3. Perform specific tasks
This Home panel and the "I want to..." menu are great for directing users to step-by-step functionality to get things done quickly. For example:

Create a mailing label report
Print a map

Welcome to the Geocortex Viewer for HTML5

The Geocortex Viewer for HTML5 is a configurable mobile viewer framework. For evaluation purposes, this app showcases available tools and features.

Among other things, you can:

1. Search for information
Find features and other results (e.g. search "ville").

2. Explore the toolbar
The toolbar is an optional element that provides numerous capabilities for more specialized, web-GIS style applications..

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 **151** 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 **47**.

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 Manager.

  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 **48** 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.

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 48

14.13 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 47.

Settings

The Instant Search page has the following settings:

- **Search Hints:** When this check box 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 check box is selected.
  When the Search Hints check box 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 check box. To rank search results only by relevance to search terms, uncheck this check box. By default, this check box 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.

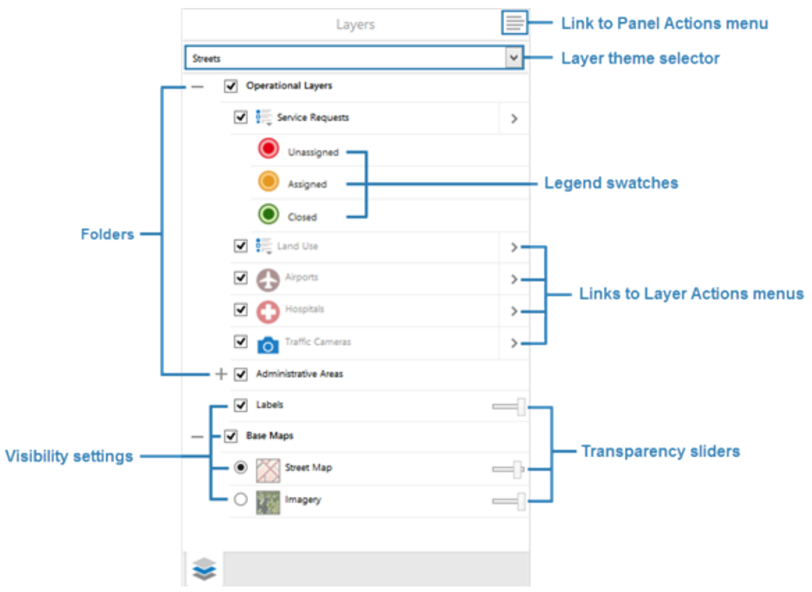
Before you leave the Instant Search page:

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.
   See **About the Live Preview** on page **48** for instructions.
3. To save your changes, click **Save Site**.

14.14 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 base maps 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

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 **47**.

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.

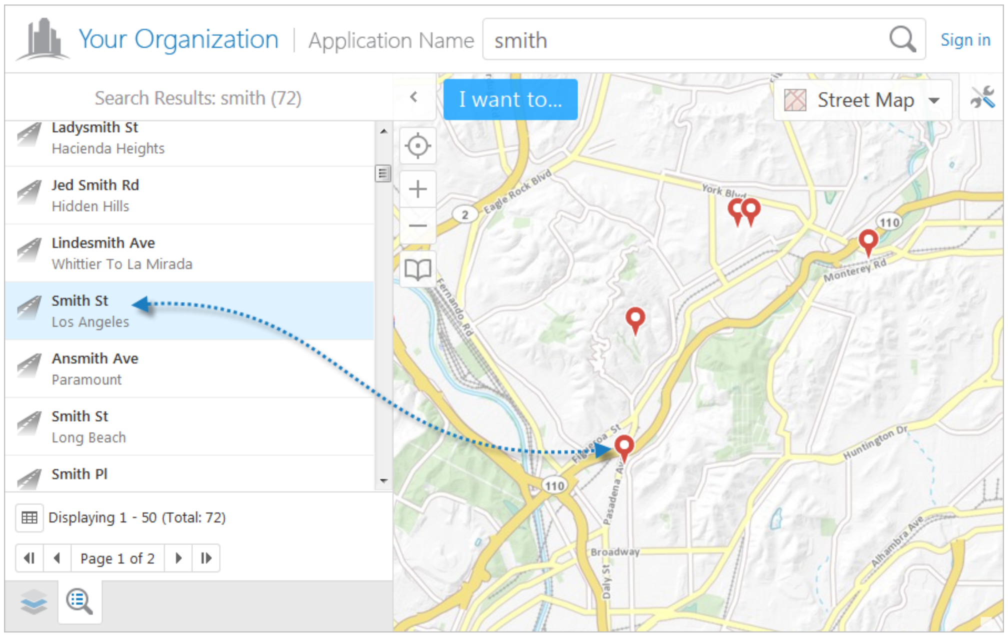
- **Show Swatches For Visible Layers Only:** Select this checkbox if you want to only display legend swatches for layers that are visible; otherwise, clear this checkbox. It is unchecked by default.

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 **48** for instructions.
3. To save your changes, click **Save Site**.

14.15 Configure Pushpins

The Pushpins page in Manager has settings that control the appearance of pushpins that appear for each feature on the current page of the search results. The pushpin is placed in the center of each feature, with a specified offset. Clicking the pushpin displays a map tip, if configured.



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.

Although it is possible to configure pushpins for the Handheld interface, doing so will have no effect since pushpins are not currently supported by the Handheld interface.

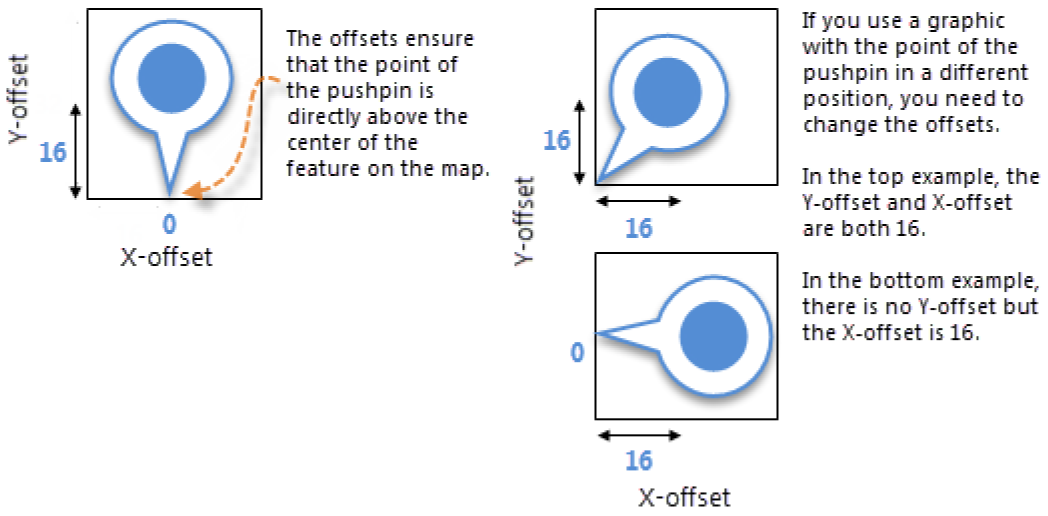
For more information, see **About Configuring Multiple Interfaces** on page **47**.

Settings

The Pushpins page has the following settings:

- **Enabled:** Check this check box if you want pushpins to appear for each feature of the search results; otherwise, uncheck it. It is checked by default.
- **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.


Y-offset
16
0
X-offset
The offsets ensure that the point of the pushpin is directly above the center of the feature on the map.
Y-offset
16
16
0
16
X-offset
If you use a graphic with the point of the pushpin in a different position, you need to change the offsets.
In the top example, the Y-offset and X-offset are both 16.
In the bottom example, there is no Y-offset but the X-offset is 16.

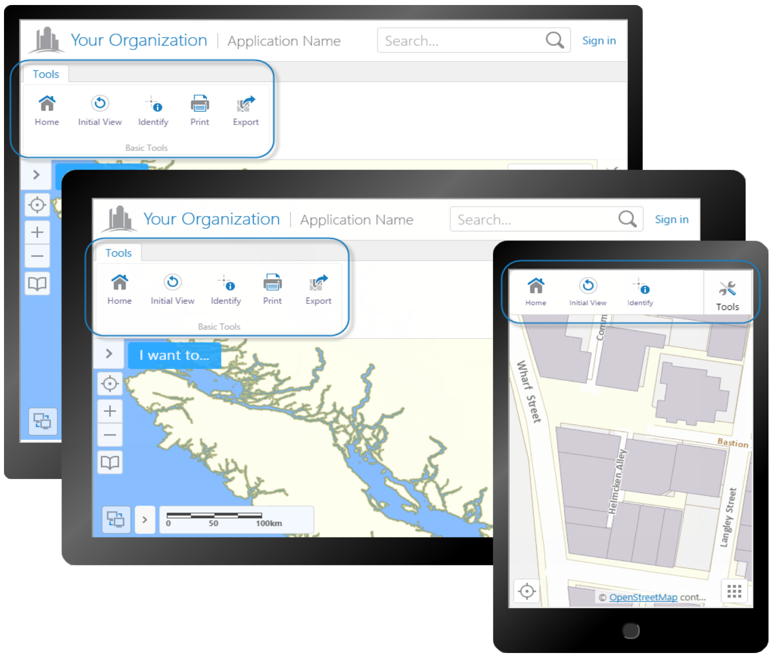
Before you leave the Pushpins page:

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.
   See **About the Live Preview** on page **48** for instructions.
3. To save your changes, click **Save Site**.

14.16 Configure the Toolbar

By default, a toolbar has a few basic tools in it to navigate the map, identify features, and print the map. As of version 2.4, 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.



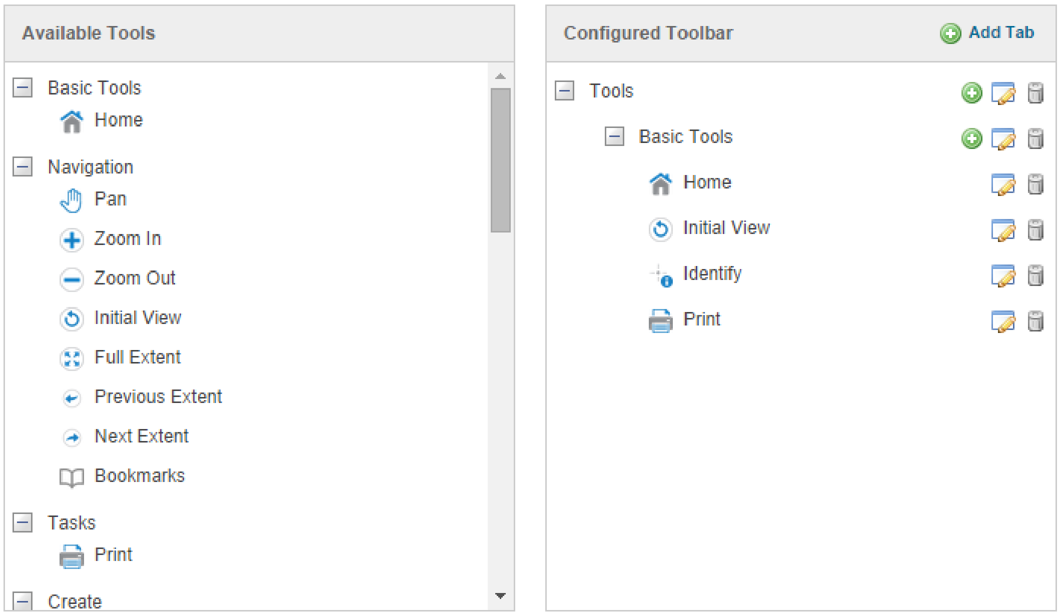
The Tabbed Toolbar in the Desktop (back) and Tablet (middle) interfaces; and the Compact Toolbar in the Handheld interface

When the user launches the viewer, the toolbar is initially closed by default. To open the toolbar, the user clicks the icon 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.

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.

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 .



Drag and drop tools to customize the toolbar

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/).

The toolbar is optional. You can turn it off using the Remove Toolbar function.

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 **47**.

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 once in the Compact Toolbar, type a number in **# of Tools to Display**.

To open the toolbar when the viewer starts:

1. Check the **Open Toolbar by Default** checkbox.

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.

To limit the number of tools displayed at once 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 disable or re-enable tool stickiness:

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.

To add a tab to the 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 **48** 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 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. In the **Layout** drop-down list, select the layout size for the group: either **Small** or **Large**.

   In the Desktop and Tablet interfaces, the **Small** Layout causes items in the group to have smaller icons, stacked in pairs. In the Handheld interface, the icons are always large, regardless of the Layout setting.

4. Leave the **Multitool** checkbox cleared.

5. Click **OK**.

   The group is added as the last group in the tab.

6. 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:

  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 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 **269**.

- **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 **269**.

3. Click **OK**.

   The button is added as the last button in the group.

4. 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.

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:
   - **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 **269**.

- **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 **269**.

- **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 **269**.

- **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 **269**.

- **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 **296**.

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 **269**.

- **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 **48** for instructions.
3. To save your changes, click **Save Site**.

See Also...

About User Interface Text on page **48**

14.17 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 **47**.

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)**.

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^2 (ft^2)**, **Yard^2 (yd^2)**, **Meter^2 (m^2)**, **Kilometer^2 (km^2)**, **Mile^2 (mi^2)**, **Nautical Mile^2 (NM^2)**, **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 **48** for instructions.
3. To save your changes, click **Save Site**.

See Also...

Configure the Toolbar on page **79**

14.18 Configure Insight Integration

Geocortex Insight collects data about and reports on GIS infrastructure. If you do not use Geocortex Insight, skip this page.

If you are running Geocortex Insight, you must configure your Viewer applications to send data to the Client Relay Collector in Insight 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 Insight Integration page in Manager enables you to configure the settings required to send this data to Insight.

To open an HTML5 viewer's Insight Integration page in Manager:

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

Before you configure settings on the Insight 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 Insight 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 **47**.

Settings

The Insight Integration page has the following settings:

- **Enabled**: When this checkbox is selected, the Viewer sends data to Insight. If you do not use Insight, clear this checkbox. By default, this checkbox is cleared.
- **Data Relay URI**: The URI to the Insight Client Relay Collector responsible for collecting data.

Before you leave the Insight Integration page:

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.
   See **About the Live Preview** on page **48** for instructions.
3. To save your changes, click **Save Site**.

14.19 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, see the Geocortex website or 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 **47**.

Settings

The Optimizer Integration page has the following settings:

- **Enabled**: When this check box is selected, the Viewer sends data to Optimizer. If you do not use Optimizer, clear this check box. By default, this check box 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 **48** for instructions.
3. To save your changes, click **Save Site**.

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 **96**. All views have the properties listed in **Common Settings for Views** on page **96**. All view models have the properties listed in **Common Settings for View Models** on page **97**.

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 **101** 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 **43**

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 **95**.
- **`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 **288** 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 **95**.

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 **95**.

15.4 Accessibility Module

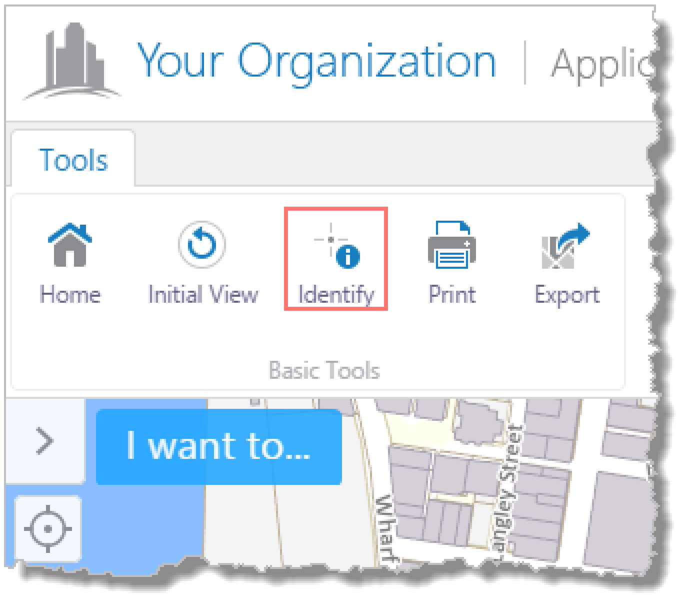
The Accessibility Module implements accessibility features that make the HTML5 Viewer easier to use for people with disabilities. There are two aspects to accessibility support in the HTML5 Viewer that users can use:

- **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`.
- **`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`.

- **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:** (Desktop and Tablet interfaces only) To display the Accessibility Button that opens the Accessibility Panel, set to `true`; otherwise, set to `false`. The default is `true`.
  - **content:** The encoded HTML content of the Accessibility Panel.
  - **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 **48**.

See Also...

Accessibility on page **253**

Configure Accessibility on page **53**

About User Interface Text on page **48**

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 **288**.
- **`overrideNativeAlert`:** When `overrideNativerAlert` is set to `true`, the browser's native alert method is overridden. By default, `overrideNativerAlert` is `true`. Calling the `alert` method with `message`, `title`, and `callback` invokes the custom alert.

Views

The Alert Module does not have any views.

View Models

The Alert Module does not have any view models.

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 **288**.

Views

The Authentication Module does not have any views.

View Models

The Authentication Module does not have any view models.

15.7 Banner Module

This module can be configured using Manager. For instructions, see **Change the Look and Feel** on page **55**.

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.
    - `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 either a web browser or 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 web browsers to scan QR codes.
  - **`jobDecoderWorkerSource`:** The `DecoderWorker.js` file from JOB (JavaScript-Only Barcode Reader). This script allows web browsers to scan barcodes.

View Models

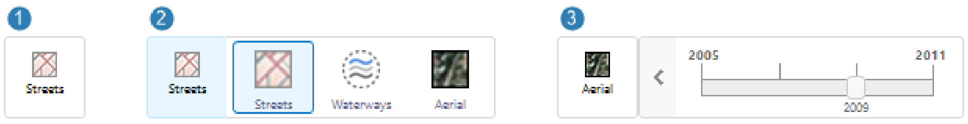
- **`BarcodeScannerViewModel`:** None

See also...

Geocortex SDK for HTML5 API Reference on page **269**

15.9 Basemap Module

The Basemap Module implements the Base Map Switcher and base map transparency sliders. The Base Map Switcher provides a quick way for users to make a single base map visible. Transparency sliders allows users to transition the map smoothly from one map service to the next. See "Base Maps" in the *Geocortex Essentials Administrator Guide* for instructions on configuring base maps and transparency slider groups.



Examples of the Base Map Switcher closed (1), open (2) and with a transparency slider (3)

In the Desktop and Tablet interfaces, the Base Map Switcher is on the map. When the user clicks the Switcher, a list of the available base maps opens. The user clicks a base map to make it visible. If the base map 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 base maps is an option in the I Want To menu. When the user taps the menu option, the list of available base maps opens so the user can change the base map.

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 Base Map 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 base maps available, the Base Map Switcher does not display. The default is `true`.
- **`BasemapSwitcherView`**: None
- **`BasemapSliderView`**: None

View Models

- **`BasemapSwitcherViewModel`**: None

Widgets

- `BasemapsListController:`
  - `grid:` The base maps and base map 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 base maps, it shrinks to the appropriate size. If the Switcher contains more base maps 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 **63** for instructions.

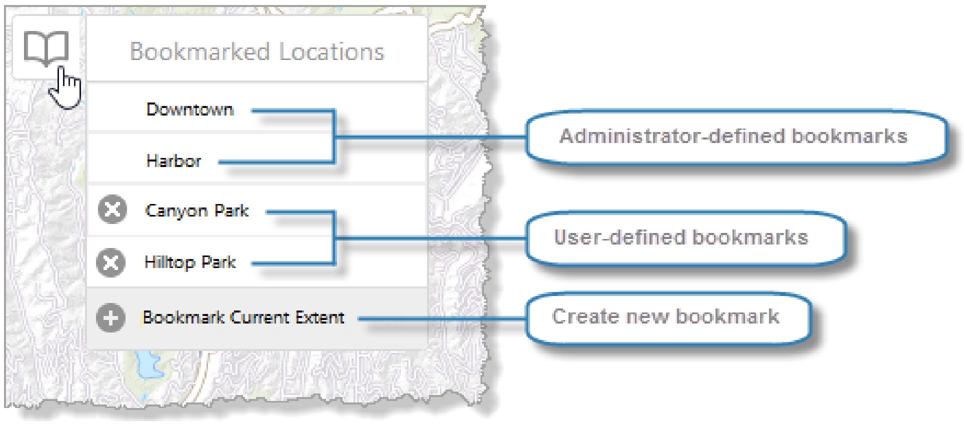
A bookmark is a shortcut that enables end users to jump the map to a particular map extent. The Bookmarks Module implements bookmarks in the HTML5 Viewer.

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 **199** 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 **269**

Configure Map Widgets on page **63**

15.11 Browser Module

This module can be configured using Manager. For instructions, see **Change the Look and Feel** on page **55**.

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 **48**.

Views

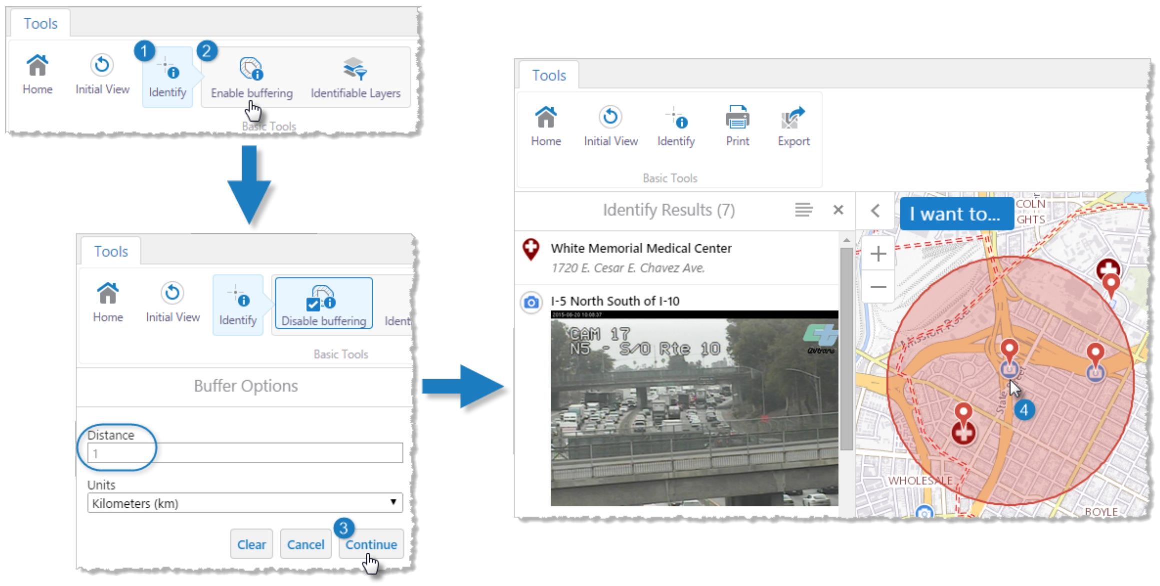
The Browser Module does not have any views.

View Models

The Browser Module does not have any view models.

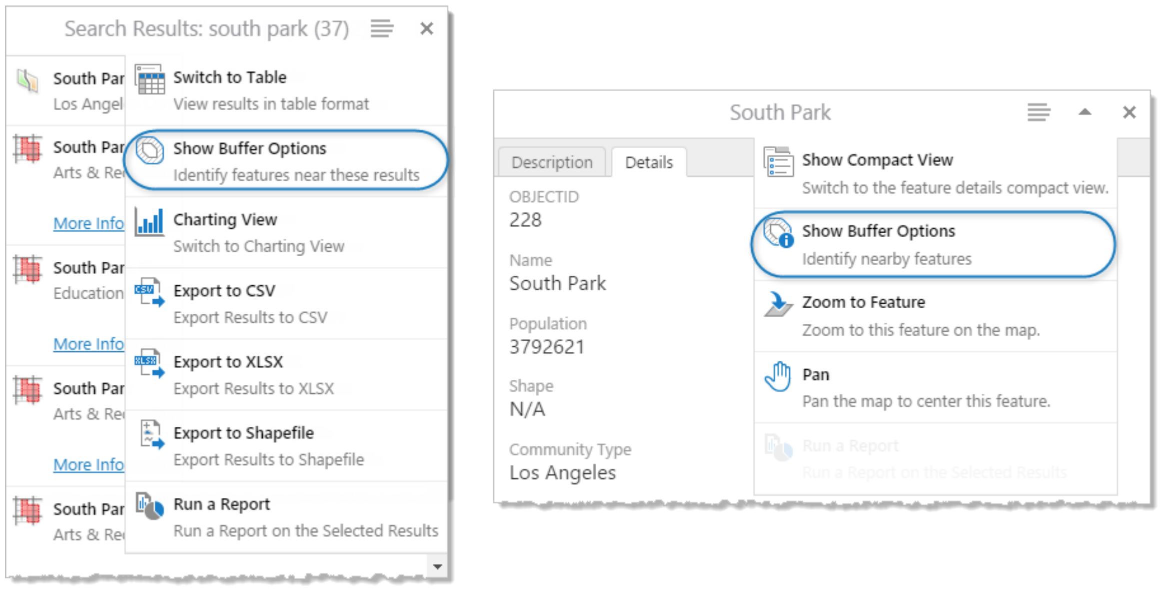
15.12 Buffer Module

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. To enable buffering, click the **Identify** tool and click **Enable Buffering** to enter buffer options. For example, you can identify features within a kilometer radius of a point you click on the map.



The user clicks Identify 1, Enable Buffering 2, selects buffer options 3, and clicks the map to identify the surrounding area 4

You can also apply a buffer to identify features surrounding all of those in the results list or a particular feature by selecting **Show Buffer Options** from the Panel Actions Menu.



OBJECTID
228

Name
South Park

Population
3792621

Shape
N/A

Community Type
Los Angeles

Identify features near those in the results list (left); identify features near a particular feature

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 a single command: `CloseDataFrame`.
    - `BufferingErrorBehavior`: A behavior that runs an array of commands when a buffering error occurs. By default, this includes a single command: `OpenDataFrame`.

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`**:**
    - `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 **91**

Identify Module on page **149**

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.

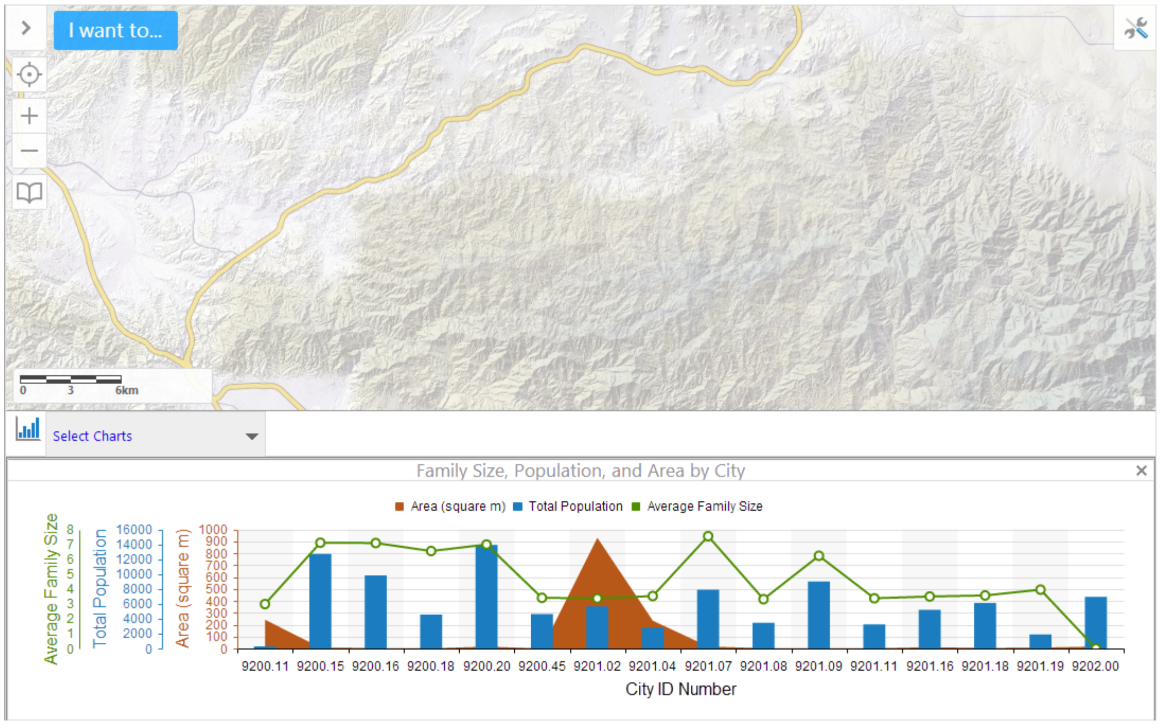


Chart example

Charts are configured in Manager. For more information on how to configure charts, refer to the *Geocortex Essentials Administrator Guide*.

NOTE It is also possible to display charts for a single feature. For more information, see **FeatureDetails Module** on page **129**.

Configuration Properties

NOTE 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 a single 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 advanced 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.essentialsHtmlViewer.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 **129**

15.14 ClusterLayers Module

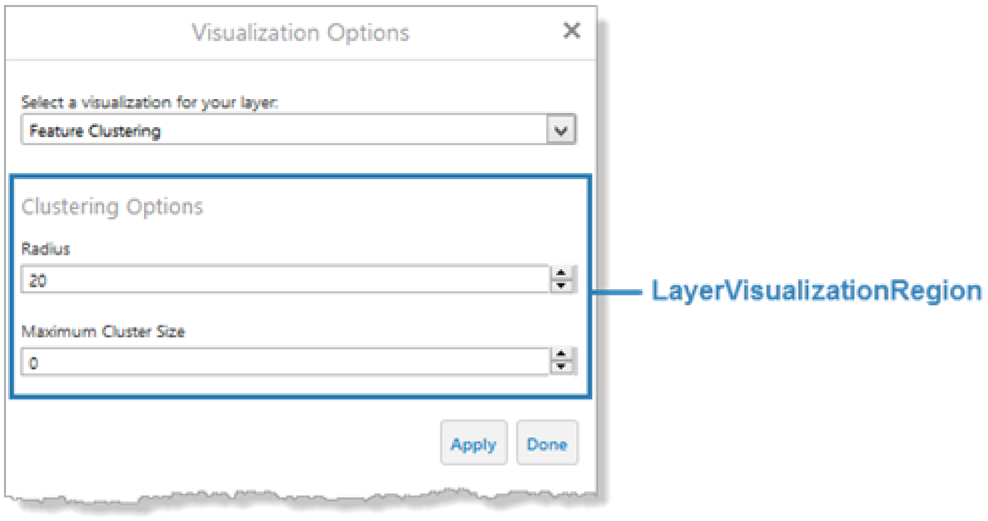
The ClusterLayers Module works with a number of other modules to implement clustering. For information about configuring clustering for a layer, refer to the *Geocortex Essentials Administrator Guide*.

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`.


Visualization Options

Select a visualization for your layer: Feature Clustering

Clustering Options

Radius: 20

Maximum Cluster Size: 0

Configuration Properties

Module

None

Views

- `ClusterLayerView`: None

View Models

- `ClusterLayerViewModel`: None

See Also...

Menu Module on page **197**

Results Module on page **213**

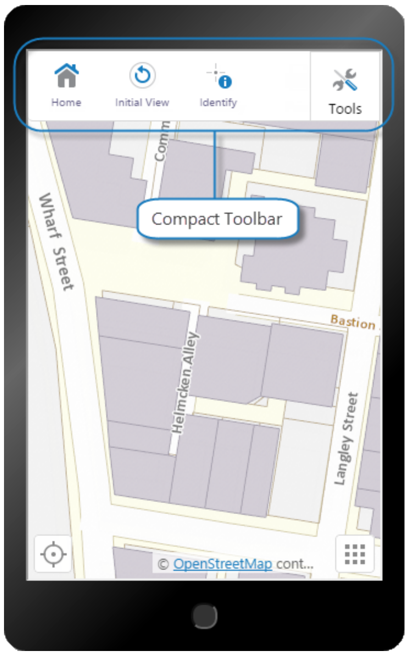
Visualization Module on page **246**

15.15 CompactToolbar Module

The toolbar can be configured using Manager. See **Configure the Toolbar** on page **79** 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 **296**.

  - `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 **269**.
- `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 **269**.
- `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`.

NOTE 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 **269**.
- **`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 **269**.
- **`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 **269**.
- **`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 **48**

TabbedToolbar Module on page **235**

15.16 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 **288**.

Views

The Confirm Module does not have any views.

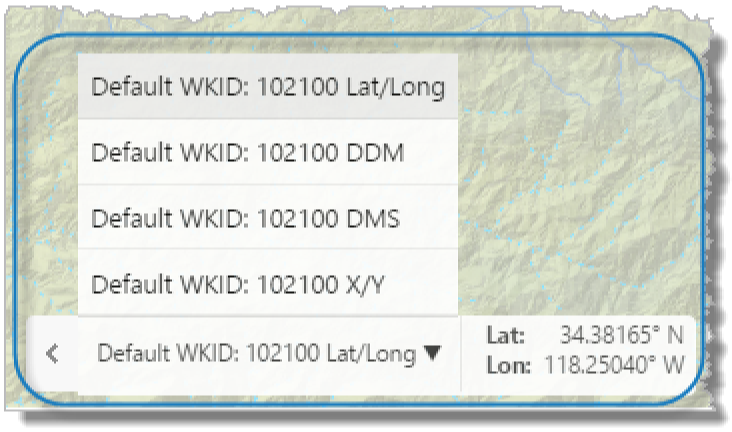
View Models

The Confirm Module does not have any view models.

15.17 Coordinates Module

This module can be configured using Manager. See **Configure Map Widgets** on page **63** 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`.
  - **`useBasemapCoordinates`:** To include the base map'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)

15.18 Editing Module

The Editing Module implements editing of 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.

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`.

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. 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 `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.

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 advanced 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 **102**.
  - `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 **102**.

    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 **48** for more information on using text keys.
    - `command`: The command that the tool runs.

      For a list of commands, see 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 **48** for more information on using text keys.

- **`command`:** The command that the tool runs.
  For a list of commands, see 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 **137**

Menu Module on page **197**

Offline Module on page **201**

Tools Module on page **244**

About User Interface Text on page **48**

15.19 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*.

If you want exported images to open automatically for the user, select the Automatically Open URLs check box on the viewer's Application page in Manager. If the user's browser blocks pop-ups, the user may have to give permission for the file to open. If you clear the Automatically Open URLs check box, users must click a button to download the file, and then open the file.

NOTE 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 **269**

15.20 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.

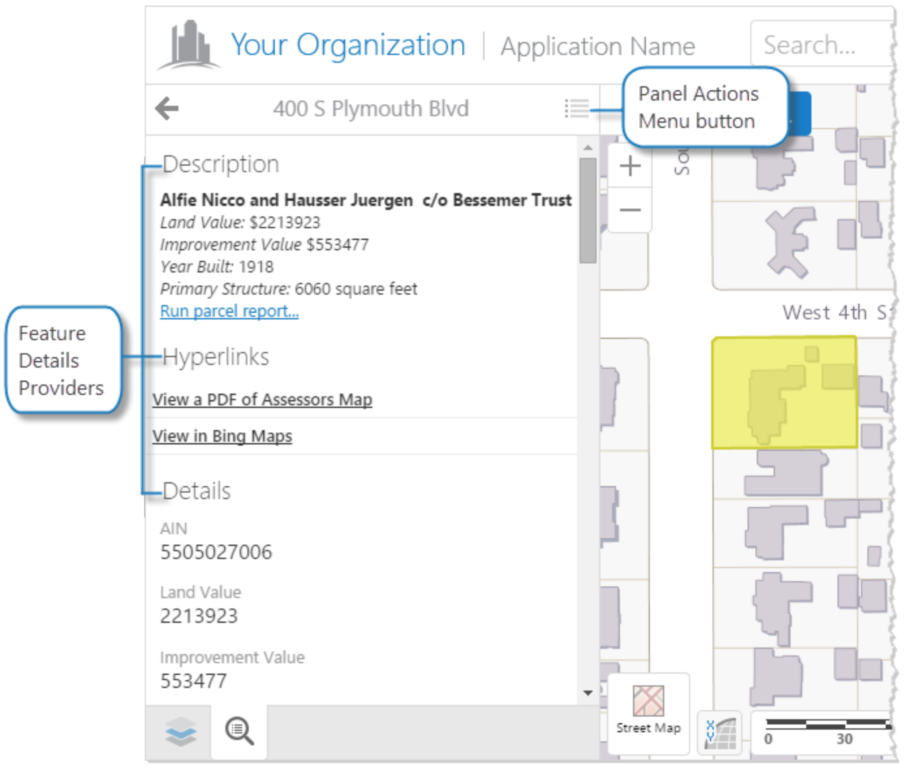
The HTML5 Viewer has the following feature details providers:

- **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 *About Charting* in the *Feature Layers* section of the *Geocortex Essentials Administrator Guide*.

- **Attachments:** Lists the feature's attachments.
- **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.



400 S Plymouth Blvd

Description

Alfie Nicco and Hausser Juergen c/o Bessemer Trust
Land Value: $2213923
Improvement Value $553477
Year Built: 1918
Primary Structure: 6060 square feet
Run parcel report...

Hyperlinks

View a PDF of Assessors Map
View in Bing Maps

Details

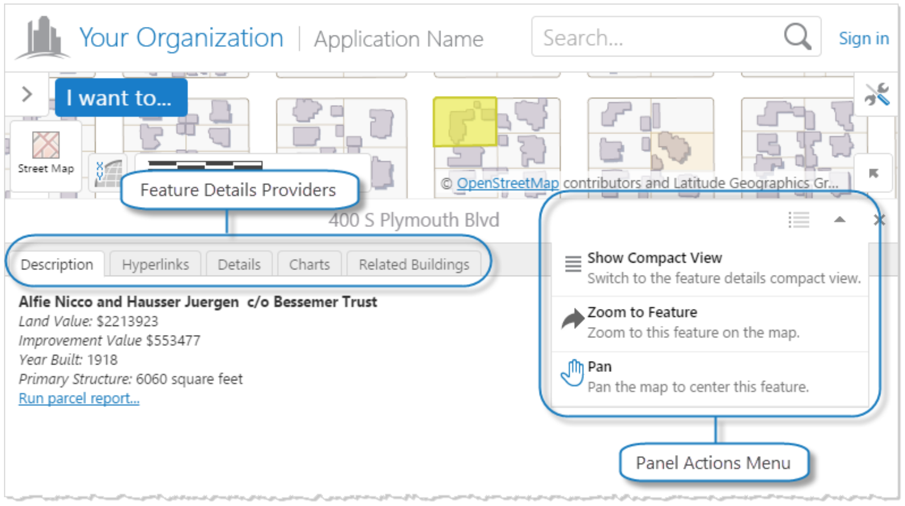
AIN
5505027006

Land Value
2213923

Improvement Value
553477

Example of feature detail providers in the Compact View

The Expanded View displays the different types of information in separate tabs, as illustrated below.


Your Organization
Application Name
Search...
Sign in
I want to...
Street Map
Feature Details Providers
© OpenStreetMap contributors and Latitude Geographics Gr...
400 S Plymouth Blvd
Description
Hyperlinks
Details
Charts
Related Buildings
Show Compact View
Switch to the feature details compact view.
Zoom to Feature
Zoom to this feature on the map.
Pan
Pan the map to center this feature.
Panel Actions Menu

Alfie Nicco and Hausser Juergen c/o Bessemer Trust
Land Value: $2213923
Improvement Value $553477
Year Built: 1918
Primary Structure: 6060 square feet
Run parcel report...

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 advanced 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 **288**.

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 **108**

Menu Module on page **197**

Regions on page **288**

15.21 FeatureLayer Module

The FeatureLayer Module implements support for feature layers in the HTML5 viewer.

The Editing and Offline modules, which implement editing and offline editing of features, depend on the FeatureLayer Module—in order for features to be editable, they must belong to a feature layer that has editing turned on.

The FeatureLayer Module has different views for the different parts of the viewer where editable features appear: `FeatureLayerDetailsView` and `FeatureLayerListView`.

Configuration Properties

Module

None

Views

- **`FeaturelayerDetailsView`**: None
- **`FeaturelayerListView`**: None

View Models

- **`FeatureLayerViewModel`**:
  - **`showWhereClause`**: If set to `true`, the user can enter a where clause in a text box on the Sync Settings panel. The where clause allows the user to filter the edits that are synchronized to the server. The default is `false`.

See Also...

Editing Module on page **124**

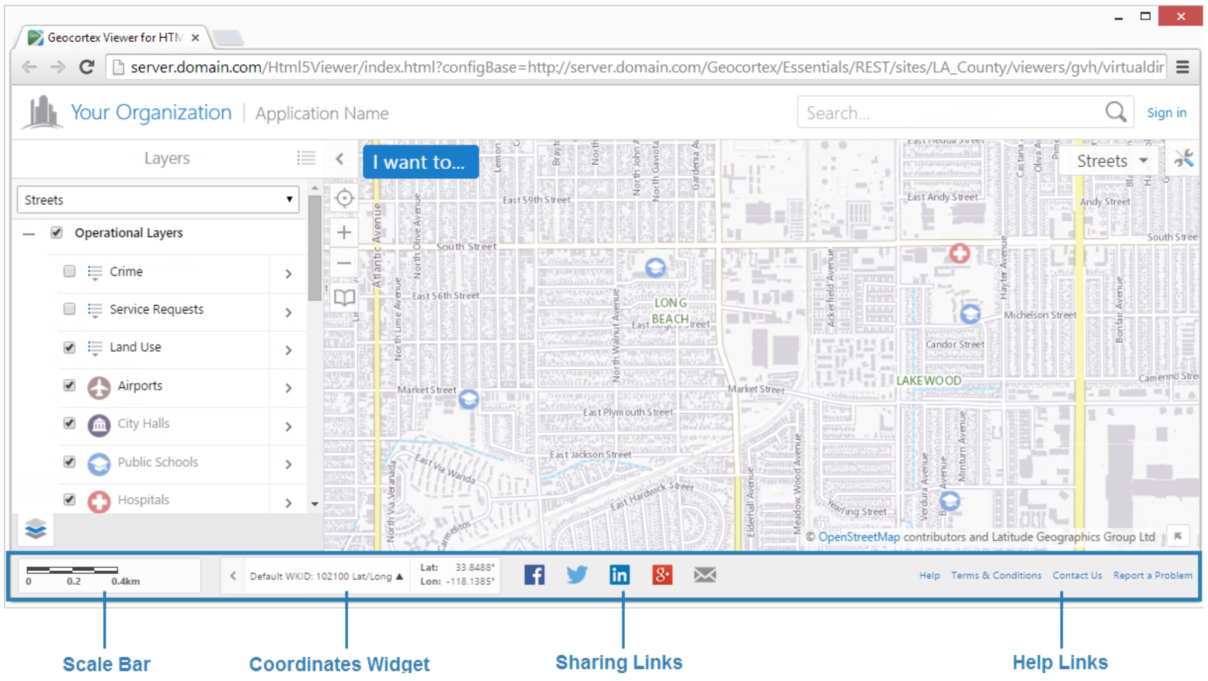
Offline Module on page **201**

15.22 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 **48** 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.
        - **`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.
        - **`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.

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 `backgoundColor` 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.

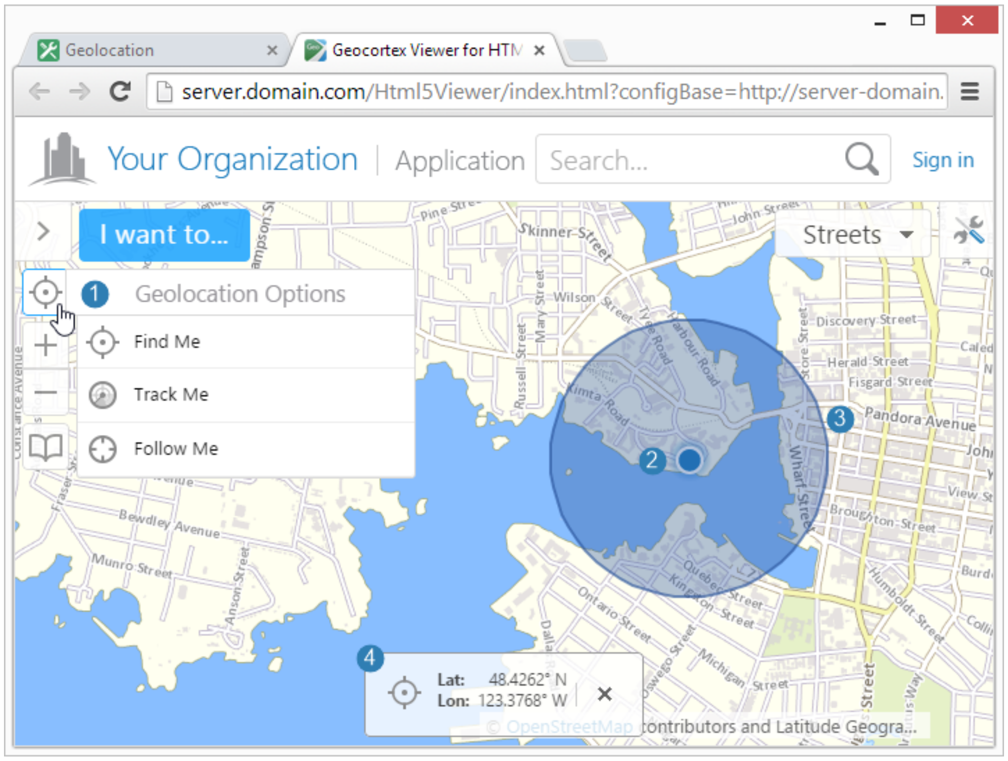
15.23 Geolocate Module

Most of the Geolocate Module's properties can be configured using Manager. For instructions, see **Configure Geolocation** on page **68**.

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)

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 **52**.

Internet Explorer 8 does not support geolocation.

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.
    - `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 **68**

Configure Application-Wide Settings on page **52**

Navigation Module on page **199**

15.24 HeatMaps Module

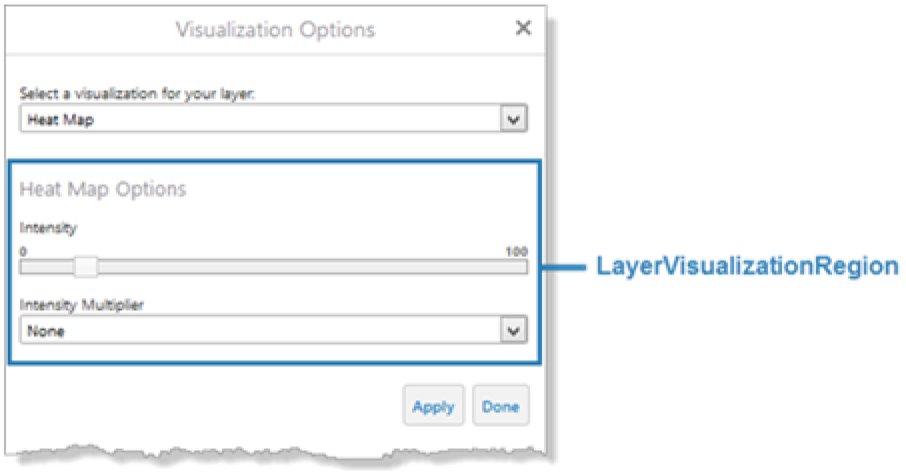
Heat mapping is a data visualization that represents the density of features on the map. The HeatMaps Module works with a number of other modules to implement heat maps. For information about configuring heat maps for a layer, refer to the *Geocortex Essentials Administrator Guide*.

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, `HeatMapsView`. `HeatMapsView` 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*.

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.

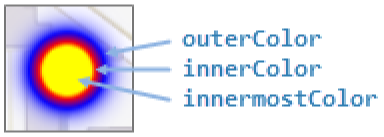
  The value that you configure here is used for layers whose Feature Heat Maps settings are not configured. For information on configuring the settings for a particular layer, refer to the *Geocortex Essentials Administrator Guide*.

  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

- `HeatMapsView`**:** None

View Models

- `HeatMapsViewModel`**:** None

See Also...

Menu Module on page **197**

Visualization Module on page **246**

15.25 Highlight Module

The Highlight Module makes it possible to configure the fill and border colors used for highlighting 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.

Configuration Properties

Module

- **`fillColor`:** The fill color to use for highlighting. The default is `"#99ECEC3A"` (pale turquoise).
- **`borderColor`:** The border color to use for highlighting.The default is `"#FFCCCC33"` (pale rose).

Views

The Highlight Module does not have any views.

View Models

The Highlight Module does not have any view models.

15.26 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 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.

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. There are two default providers:
  - **`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 **48** for more information on using text keys.
  - **`command`:** The command that the tool runs.

    For a list of commands, see 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

- **`IdentifyOptionsView`:** None

View Models

- **`IdentifyOptionsViewModel`:** None

See Also...

MapTips Module on page **180**

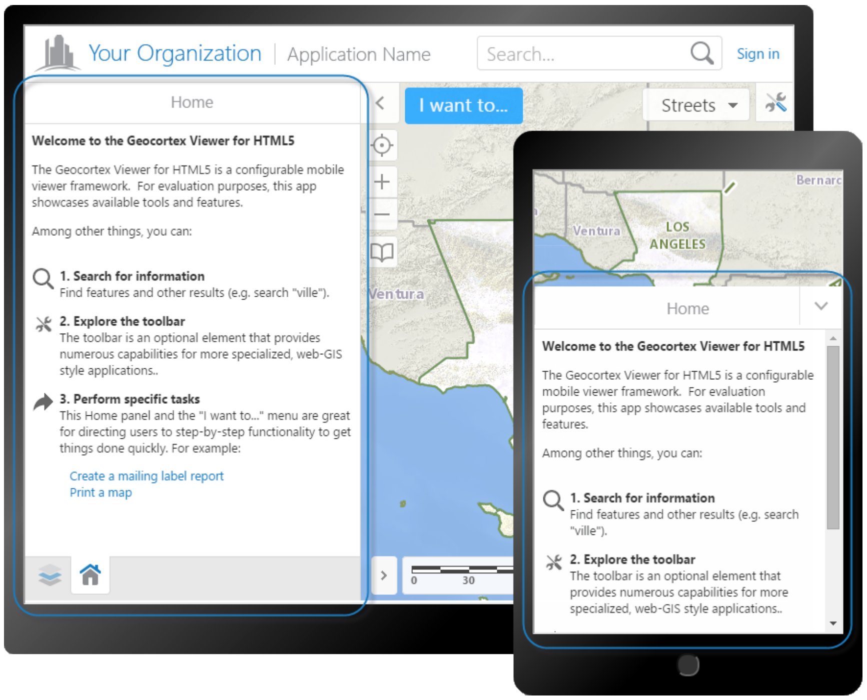
Tools Module on page **244**

About User Interface Text on page **48**

15.27 Info Module

This module can be configured using Manager. For instructions, see **Configure the Home Panel** on page **71**.

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.



Welcome to the Geocortex Viewer for HTML5

The Geocortex Viewer for HTML5 is a configurable mobile viewer framework. For evaluation purposes, this app showcases available tools and features.

Among other things, you can:

1. Search for information
Find features and other results (e.g. search "ville").

2. Explore the toolbar
The toolbar is an optional element that provides numerous capabilities for more specialized, web-GIS style applications..

3. Perform specific tasks
This Home panel and the "I want to..." menu are great for directing users to step-by-step functionality to get things done quickly. For example:

Create a mailing label report
Print a map

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.

    Manager HTML encodes the Home Panel `content` string when you edit and save the viewer configuration in Manager.
  - **`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 **48** for more information on using text keys.

    The default title is `@language-common-welcome`.

See Also...

Shells Module on page **226**

About User Interface Text on page **48**

15.28 InsightIntegration Module

The `InsightIntegration` Module implements integration between Insight and the HTML5 viewer.

The `InsightIntegration` Module collects usage and other data about the viewer and sends it to the Client Relay Collector in Insight. The `InsightIntegration` module creates a file in isolated 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 module also creates a new file and starts adding events to the new file. The module sends the zipped file to the URL that you configure for 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 Geocortex Insight.

NOTE The domains of and the HTML5 Viewer are likely different. If so, to ensure the HTML5 Viewer can send data to Geocortex Insight, you must add a proxy entry for Geocortex Insight 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/Insight`**`" matchAll="true"></serverUrl>`.

Configuration Properties

Module

- **`enabled`:** If set to `true`, the module is enabled.
- **`dataRelayUri`:** Sets the URL to the Client Relay in Insight. For example, `"http://localhost/Geocortex/Insight/Insight/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

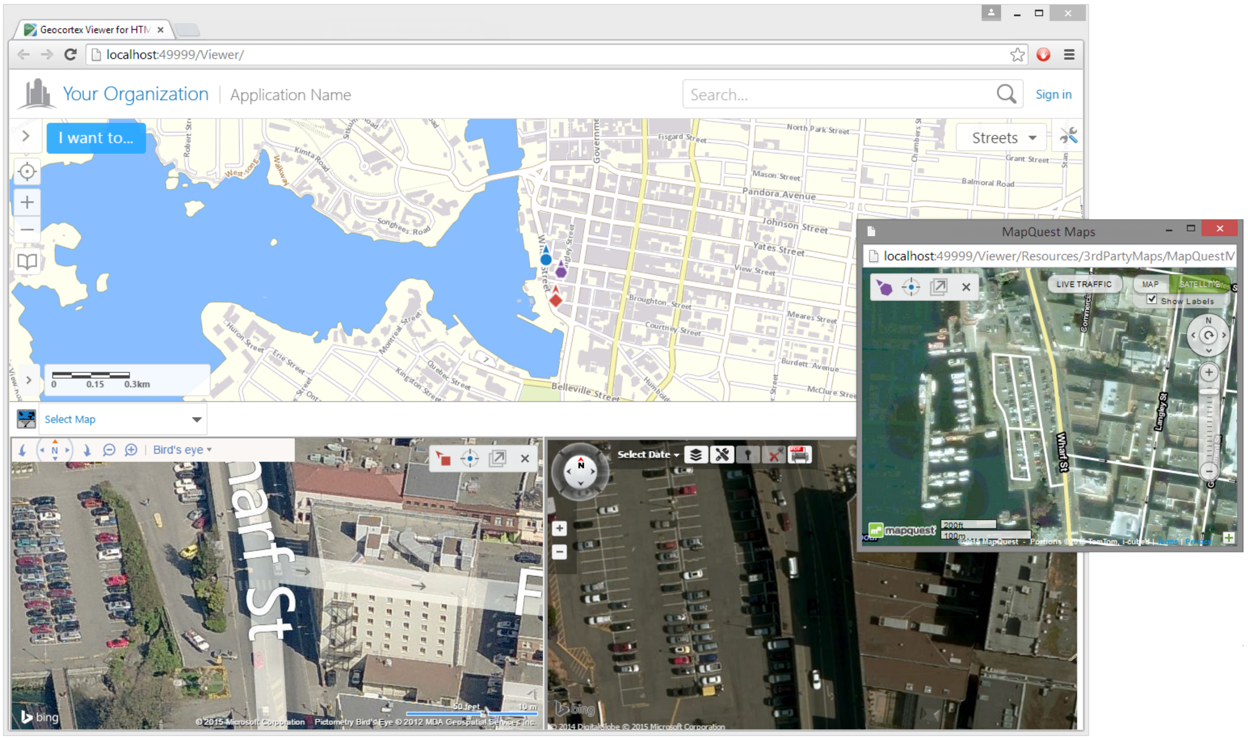
- **`ExternalComponentViewModel`:**
    - **`containerRegionName:`** The name of the container region in which to host Insight integration. The default is **`ExternalComponentRegion.`**
    - **`containerRegionType:`** The type of the container region in which to host Insight integration.. The default 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 Insight Integration on page **93**

15.29 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 9 supports integration, but does not support undocking integration panes. Internet Explorer 8 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.

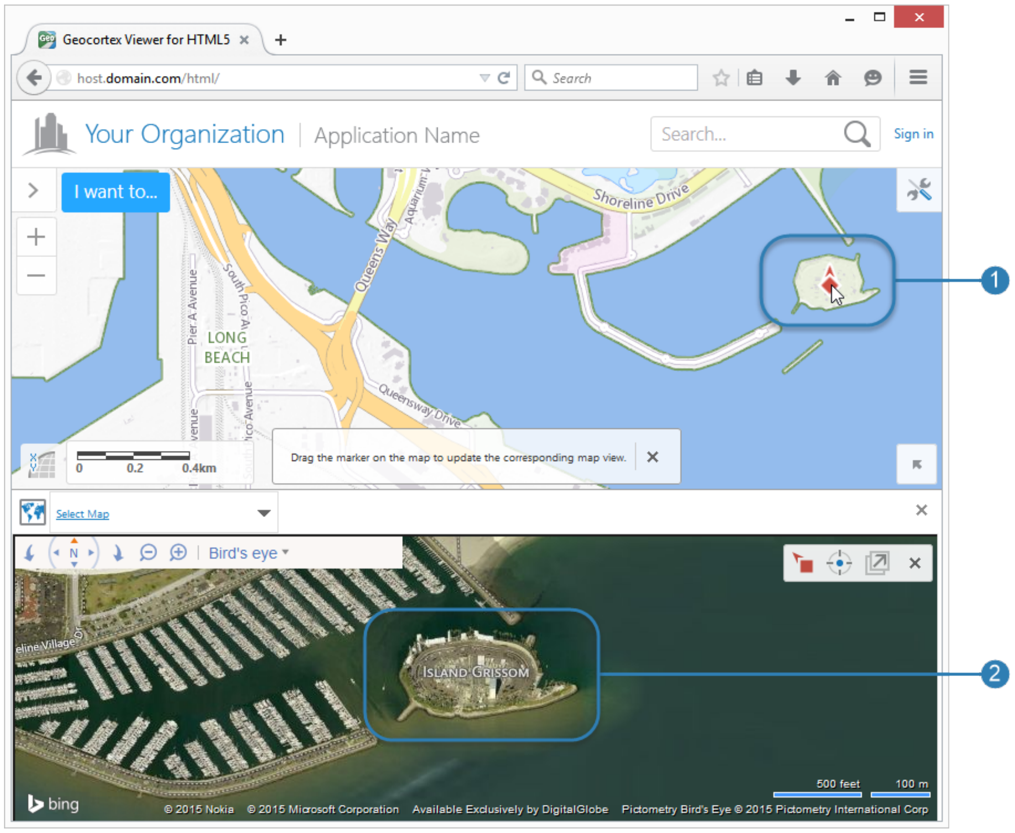
You may want to provide additional controls for interacting with an integrated application. For example, when you integrate a mapping application, you may want users to be able to center the integrated map on the viewer's map. In this case, you host a web page that points to the application that you 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 **161** and **Example 2: Integrate with Pictometry** on page **165** for instructions. As well, the Geocortex Support Center's Code Gallery includes a sample that integrates Google Street View.

Integration with 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.


Geocortex Viewer for HTML5
host.domain.com/html/
Search
Your Organization
Application Name
Search...
Sign in
I want to...
Shoreline Drive
LONG BEACH
Queensway Drive
0 0.2 0.4km
Drag the marker on the map to update the corresponding map view.
Select Map
Bird's eye
ISLAND GRISSOM
500 feet
100 m
bing
© 2015 Nokia © 2015 Microsoft Corporation Available Exclusively by DigitalGlobe Pictometry Bird's Eye © 2015 Pictometry International Corp
1
2

Dragging the marker (1) pans the integrated map so it stays centered on the marker (2)

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 interface 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.

15.29.1 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 **159**.
  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 **159** 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"
   }
]
```

15.29.2 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.

15.29.3 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.

15.29.4 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. If the viewer belongs to a secured site and SSL is enforced:
   a. Find the following line:

   ```
   <script type="text/javascript"
   src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></scri
   pt>
   ```

   b. Change the URL's protocol to `https` and add `&s=1` to the end of the URL.
   The line should look like this:

   ```
   <script type="text/javascript"
   src="https://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0&s=1"><
   /script>
   ```

6. 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.essentialsHtmlViewer.mapping.modules.integration.IntegrationModul
    e",
       "configuration": {
          "allowedOrigins": ["http://mymaps.mydomain.com"]
       }
       ...
    ```

    NOTE 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**.

15.29.5 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://mymaps.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.IntegrationModul
   e",
      "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.jsallowedOrigins` 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.

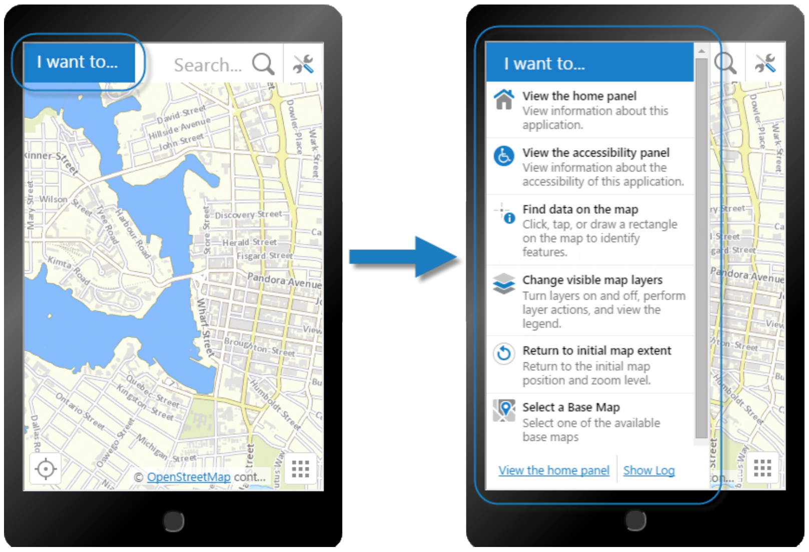
Step 4: Add the Linked Maps button to the toolbar

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**.

15.30 IWantToMenu Module

This module can be configured using Manager. For instructions, see **Configure the I Want To Menu** on page **58**.

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...

View the home panel
View information about this application.

View the accessibility panel
View information about the accessibility of this application.

Find data on the map
Click, tap, or draw a rectangle on the map to identify features.

Change visible map layers
Turn layers on and off, perform layer actions, and view the legend.

Return to initial map extent
Return to the initial map position and zoom level.

Select a Base Map
Select one of the available base maps

View the home panel | Show Log

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 setting the `IWantToMenuViewModel`'s `showMenu` property to `false` in the configuration files. Alternatively, you can clear the **Show Menu** check box on the viewer's I Want To Menu page in Manager.

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 **48** 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.
    - `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.
    - `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.
    - `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.

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 appears in the viewer. If `showMenu` is set to `false`, the viewer does not have an I Want To menu.
  - `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 **48** for more information on using text keys.

See Also...

Menu Module on page **197**

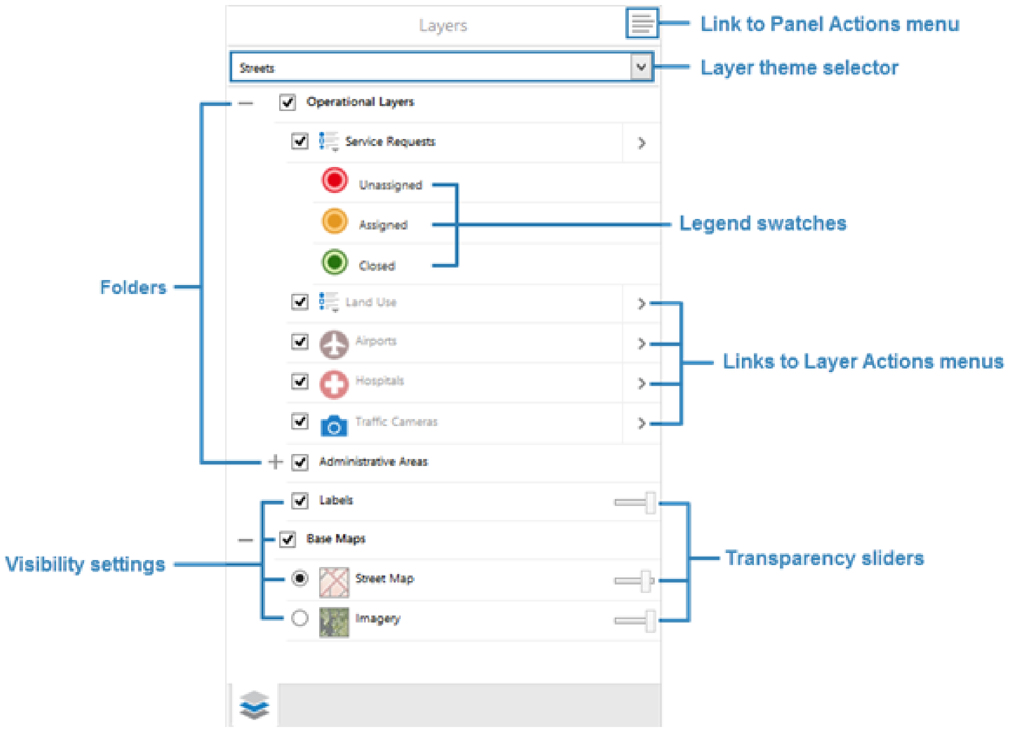
15.31 LayerList Module

As of Geocortex Viewer for HTML5 2.3, the SimpleLayerList Module no longer exists and has been replaced by the LayerList Module.

This module can be configured using Manager. For instructions, see **Configure the Layer List** on page **75**.

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 base maps 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

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.

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

See Also...

Configure the Layer List on page **75**

Menu Module on page **197**

15.32 LayerSelector Module

The LayerSelector Module implements the layer selector panel that appears when the user clicks either the **Identifiable Layers** or **Select Snapping Layers** button. This allows the user to select which layers are affected by either the **Identify** tools or **Enable Snapping** feature, respectively.

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 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 **Select Snapping Layers** button is configured in context-sensitive toolbars within the TabbedToolbar and CompactToolbar modules. The following tools support snapping: **Measure**, **Draw**, **Edit**, **Create New Feature**, and non-dragging **Identify** tools. The easiest way to configure this feature for each tool 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

- `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 a single command: `OpenDataFrame`.
    - `SelectLayersForSnappingActivatedBehavior:` A behavior that runs an array of commands when the user clicks the **Select Snapping Layers** button. By default, this includes a single command: `OpenDataFrame`.

  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 advanced developers.

Views

- `IdentifyLayerSelectorView`**:** None
- `SnappingLayerSelectorView`**:** None

View Models

- `IdentifyLayerSelectorViewModel`**:** None
- `SnappingLayerSelectorViewModel`**:** None

See Also...

Identify Module on page **149**

Snapping Module on page **232**

TabbedToolbar Module on page **235**

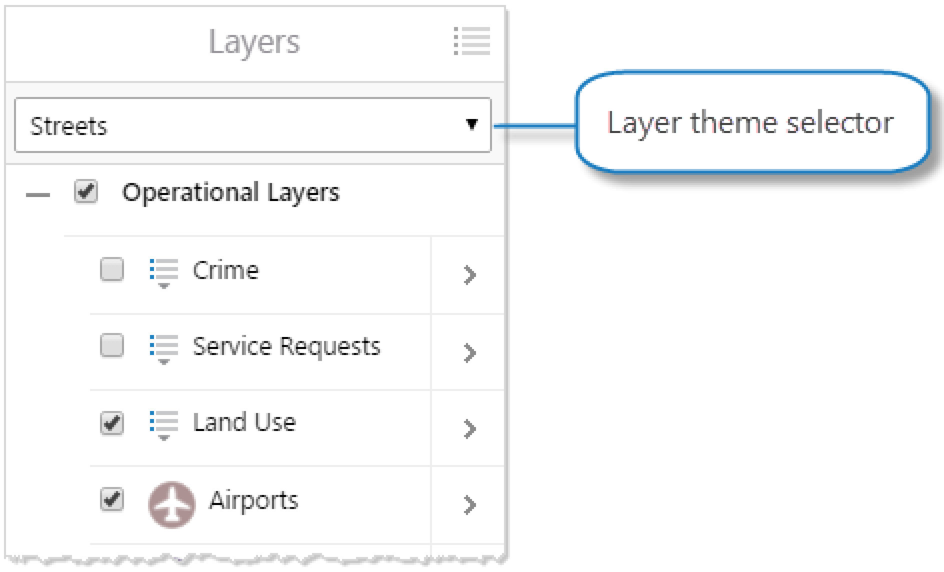
CompactToolbar Module on page **114**

15.33 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 **36**.



Layer List with the Streets layer theme selected

Configuration Properties

Module

None

Views

None

View Models

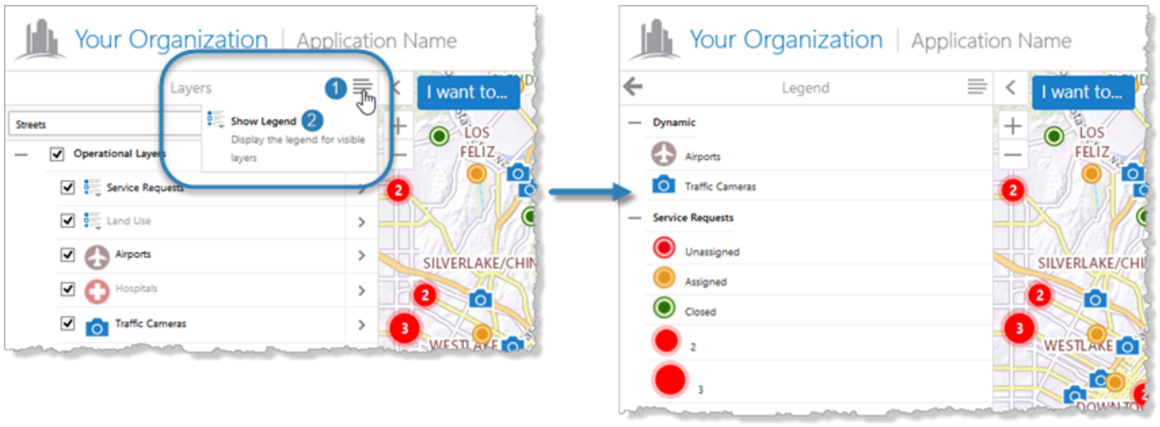
None

See Also...

LayerList Module on page **170**

15.34 Legend Module

The Legend Module implements the map legend. To show the Legend panel, open the **Layers** panel, open the **Panel Actions Menu** 1 and select **Show Legend** 2.



Open the legend

Users can also view legend swatches in the Layers panel. See **LayerList Module** on page **170** for information.

Configuration Properties

Module

None

Views

- `LegendView`: None

View Models

- `LegendViewModel`: None

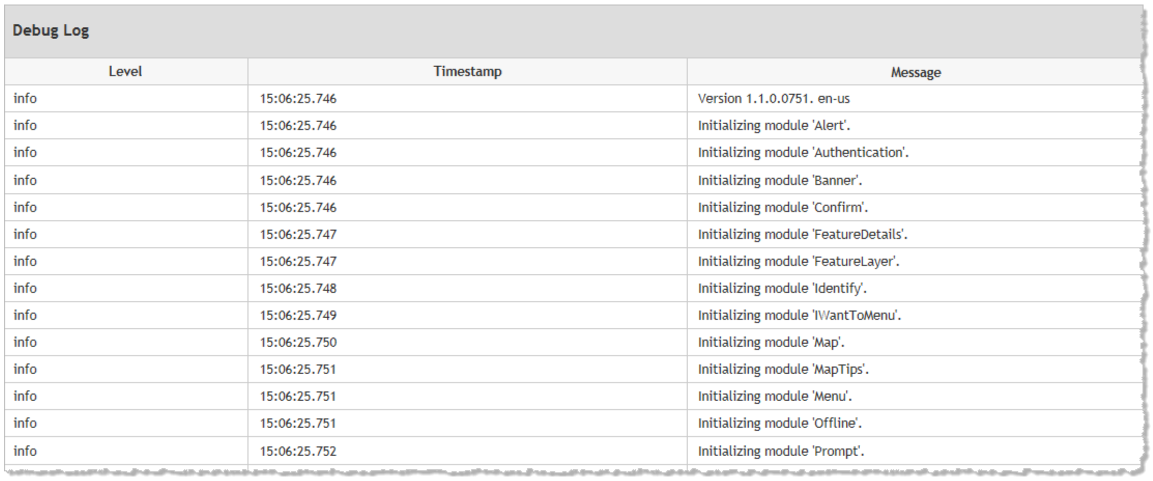
15.35 Log Module

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.

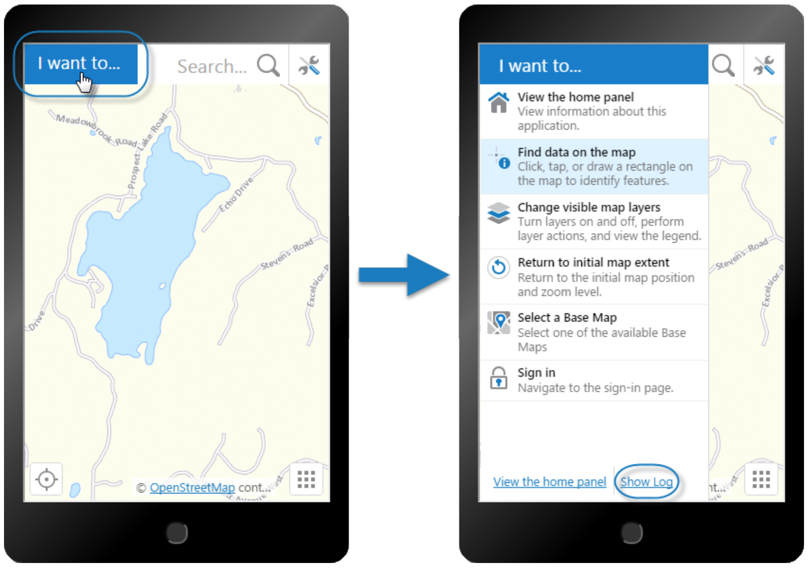


Debug Log

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**.


I want to...
Search...
I want to...
View the home panel
View information about this application.
Find data on the map
Click, tap, or draw a rectangle on the map to identify features.
Change visible map layers
Turn layers on and off, perform layer actions, and view the legend.
Return to initial map extent
Return to the initial map position and zoom level.
Select a Base Map
Select one of the available Base Maps
Sign in
Navigate to the sign-in page.
View the home panel
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

- `LogViewModel`: None

15.36 Map Module

The Map Module manages the map and map-related events.

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 two commands: `InvokeMapTip` and `ClearDefaultHighlights`.
  - **`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, no commands are configured to run.
  - **`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 a single command: `ShowMapTip`.

    Modifying the `MapOnFeatureClickBehavior` property is an advanced task. We recommend that you use the default configuration for `MapOnFeatureClickBehavior`.

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 advanced developers.

- **`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 **48** for more information on using text keys.
  - **`command`:** The command that the tool runs.

    For a list of commands, see 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`.
  - `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.

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.

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...

Tools Module on page **244**

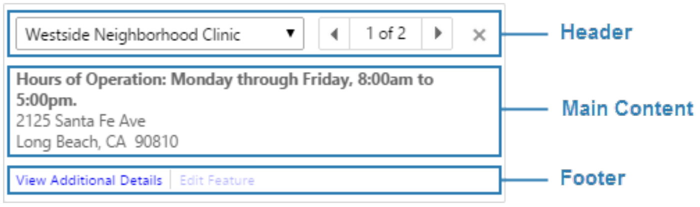
About User Interface Text on page **48**

ZoomControl Module on page **250**

15.37 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.



Hours of Operation: Monday through Friday, 8:00am to 5:00pm.
2125 Santa Fe Ave
Long Beach, CA 90810

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 **197**.

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 **213** for information.

The Identify module performs the query when a user clicks the map. See **Identify Module** on page **149** 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.

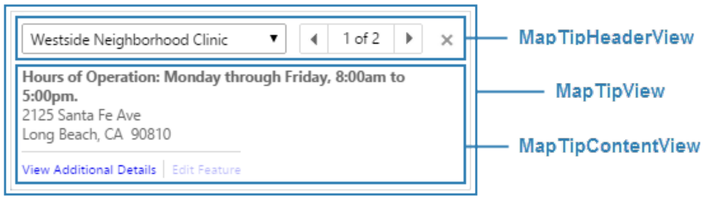
The HTML5 Viewer has the following commands for working with fixed-location map tips:

- `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—`MapTipView`, `MapTipHeaderView`, and `MapTipContentView`. Fixed-location map tips use all three of these views.

`MapTipHeaderView` contains the map tip's header. `MapTipContentView` contains the map tip's main content and footer. Both of these views appear within `MapTipView` in the viewer. The three views all share the same view model, `MapTipViewModel`.



Hours of Operation: Monday through Friday, 8:00am to 5:00pm.
2125 Santa Fe Ave
Long Beach, CA 90810

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`

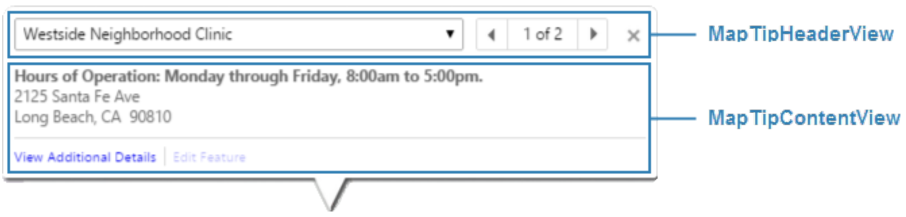
For more information about these commands, see "Commands" in the Geocortex SDK for HTML5 API Reference.

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.

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`.



Hours of Operation: Monday through Friday, 8:00am to 5:00pm.
2125 Santa Fe Ave
Long Beach, CA 90810

Views used by callout-style map tips

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 a single command: `ClearDefaultHighlights`.
  - **`MapCalloutClosedBehavior:`** A behavior that runs an array of commands when the user closes a callout-style map tip. By default, this includes a single command: `ClearDefaultHighlights`.
  - **`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 a single command: `PanToFeatureIfOutsideMapExtent`.

  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 advanced 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.

  NOTE Modifying the `webMapFeaturePresenter` property is an advanced task. We recommend that you use the default configuration for the `webMapFeaturePresenter`.

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 **269**

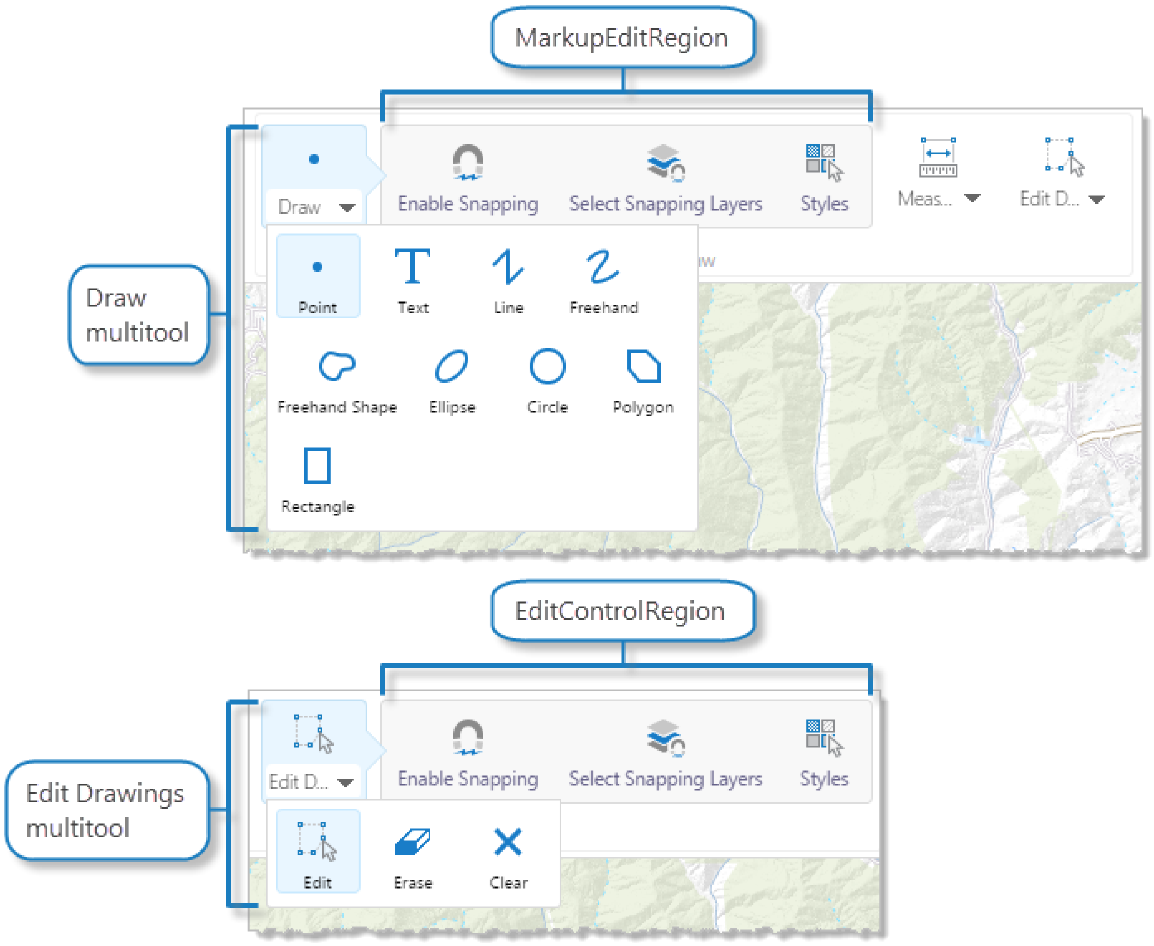
Identify Module on page **149**

Menu Module on page **197**

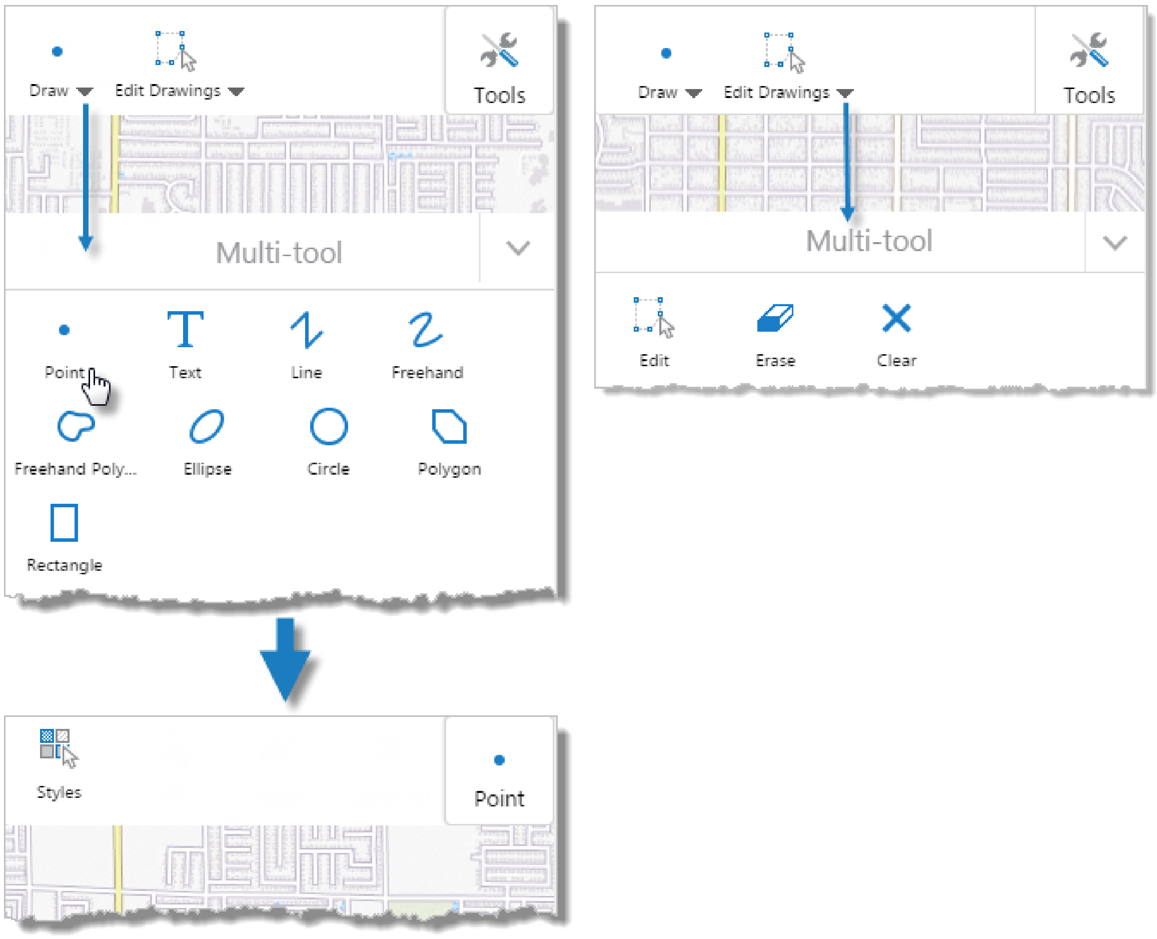
Results Module on page **213**

15.38 Markup Module

The Markup Module contains tools that allow an end user to add markup shapes and markup text to a map. You can also edit or erase existing markup.



Drawing creation tools (left) and editing tools (right) in the Desktop and Tablet interfaces



Drawing creation tools (left) and editing tools (right) in the Handheld interfaces

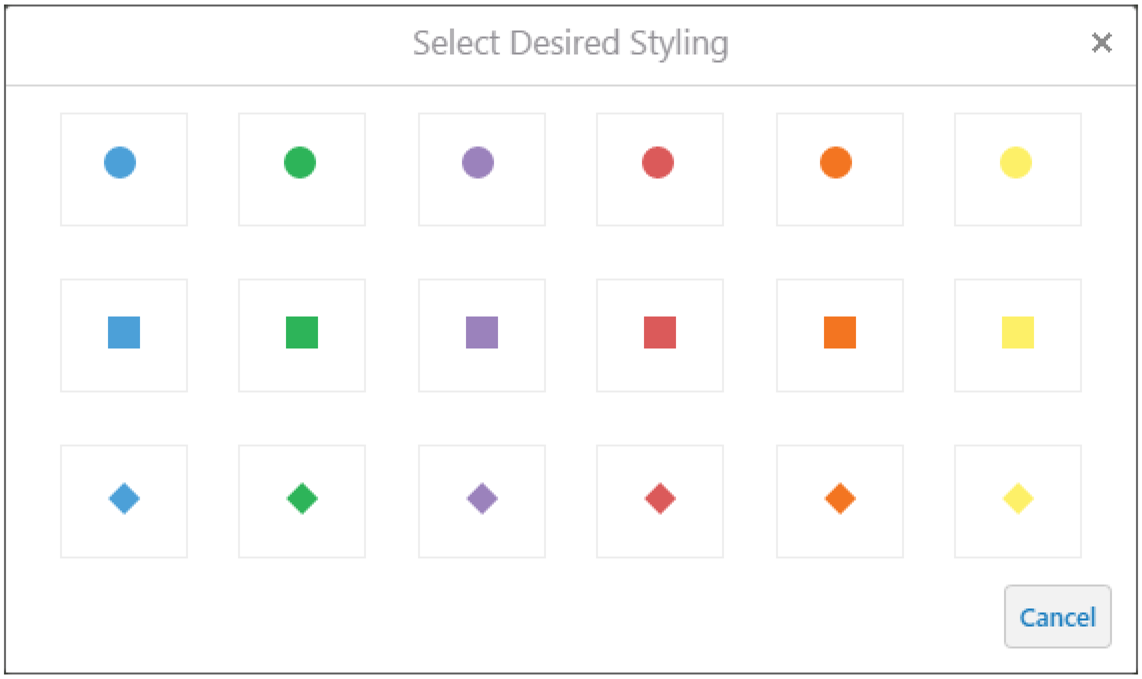
The easiest way to add drawing tools to the toolbar is to edit the viewer in Essentials Manager, and navigate to the **Toolbar** section. To add the drawing creation tools and its context-sensitive toolbar, add both the **Draw** multitool and **MarkupEditRegion** to your configured toolbar. The **Draw** multitool includes all drawing tools of various shapes, while the **MarkupEditRegion** includes snapping and styling tools.

To add the drawing editing tools and its context-sensitive toolbar, add both the **Edit Drawings** multitool and **EditControlRegion** to your configured toolbar. The **Edit Drawings** multitool includes edit, erase and clear tools, while the **EditControlRegion** includes snapping and styling tools.

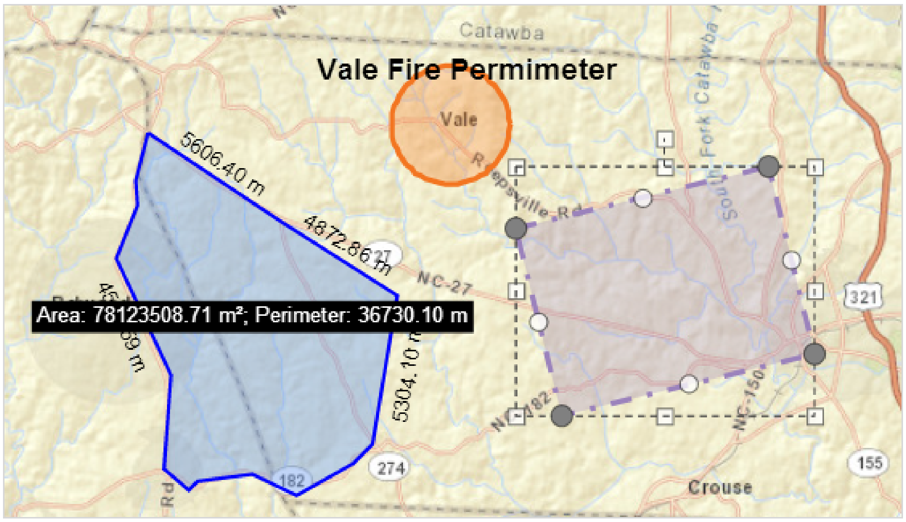
In the case of the Compact Toolbar, do not include the regions.

As of HTML5 Viewer 2.5, the **Erase** and **Clear** tools apply to both drawings and measurements.

From version 2.0, it is possible to add text as markup and also edit the shape and style of markup within the viewer. When you click the **Styles** tool, you are presented with a set of default styles to choose from.



You can edit these default styles or add your own custom styles in the Module configuration.



Different types and styles of markup

The Markup Module and the Measurement Module are integrated so if you add measurement markup to the map, you can style and edit it like other markup. When you edit measurement markup and you change the length or area of the markup, the 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 pieces of markup are stacked on top of each other, the highest markup is edited. If you want to edit markup under other markup, you need to separate them first.

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**.
    - `lineSize`: 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, **"12pt"**.

      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.
  - `customLineStyles`: 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 **79**

Measurement Module on page **189**

CompactToolbar Module on page **114**

TabbedToolbar Module on page **235**

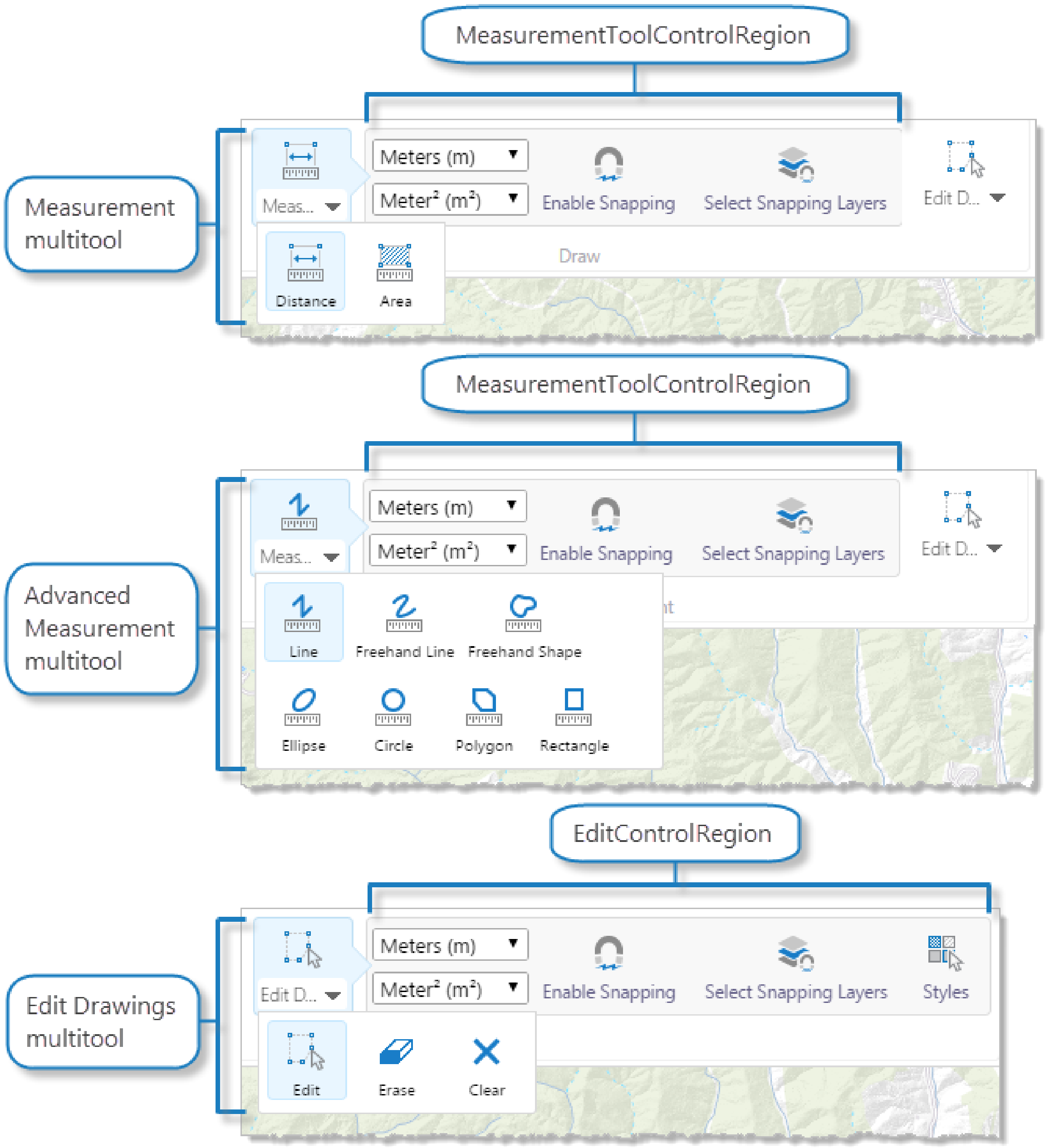
Snapping Module on page **232**

15.39 Measurement Module

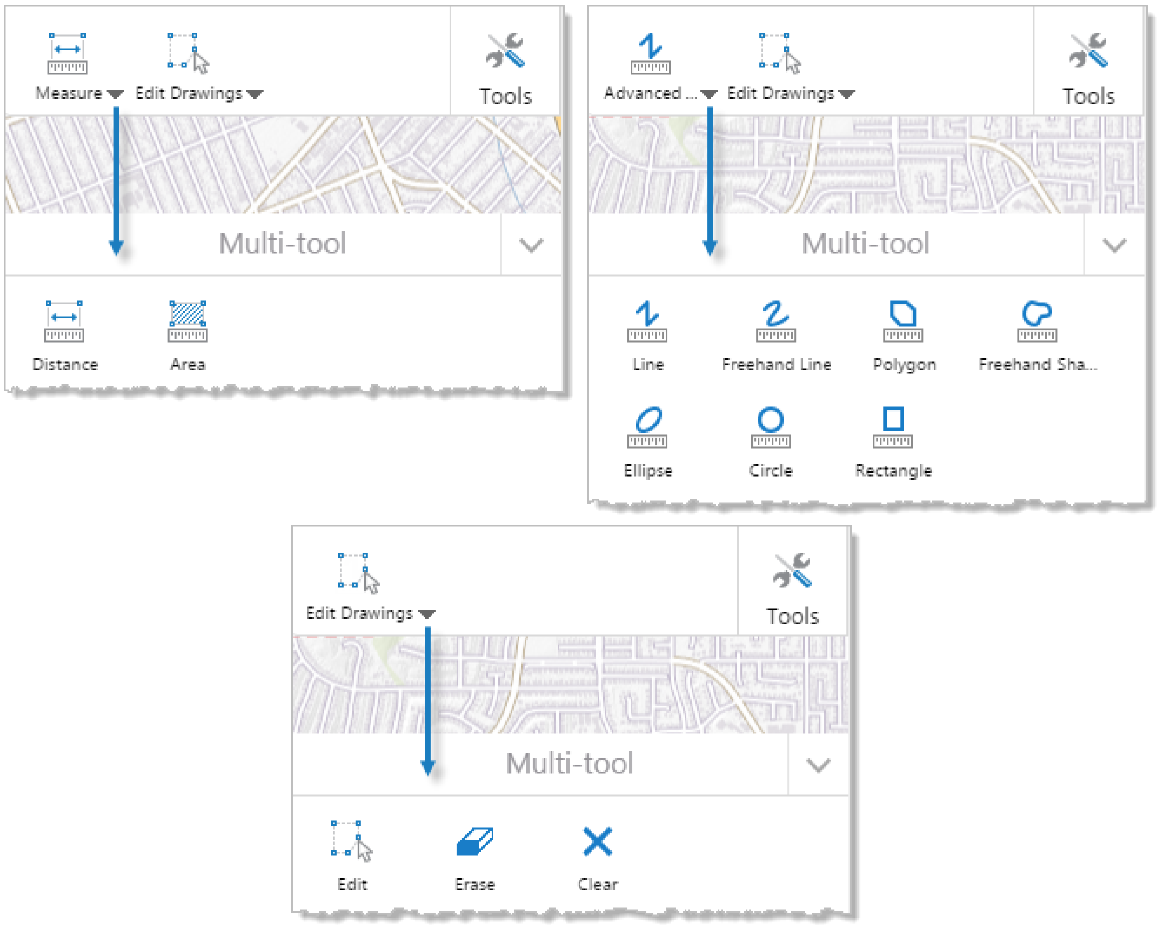
This module can be partially configured using Manager. For instructions, see **Configure Tool Behavior** on page **91**.

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), advanced measurement tools (middle), and measurement editing tools in the Desktop and Tablet interfaces



Measurement tools (left), advanced measurement tools (right), and measurement editing tools in the Handheld interface

The easiest way to add measurement tools to the toolbar is to edit the viewer in Essentials Manager, and navigate to the **Toolbar** section. To add the measurement creation tools and its context-sensitive toolbar, under the **Measure** group, add both the **Measure** multitool and **MeasurementToolControlRegion** to your configured toolbar. The **Measure** multitool includes measurement tools for both distance and area, while the **MeasurementToolControlRegion** includes measurement units, snapping and styling tools.

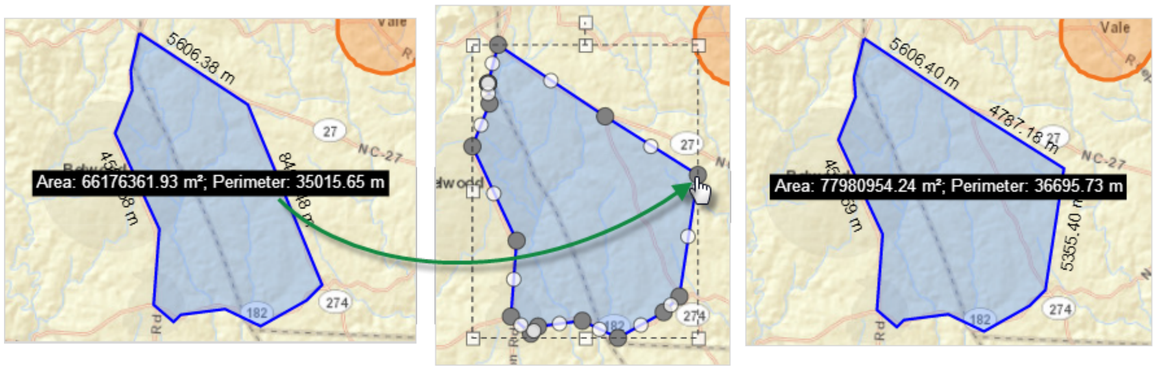
To add the advanced measurement creation tools and its context-sensitive toolbar, under the **Advanced Measurement** group, add both the **Measure** multitool and **MeasurementToolControlRegion** to your configured toolbar. The **Measure** multitool includes all measurement tools of various shapes, while the **MeasurementToolControlRegion** includes measurement units, snapping and styling tools.

To add the measurement editing tools and its context-sensitive toolbar, add both the **Edit Drawings** multitool and **EditControlRegion** to your configured toolbar. The **Edit Drawings** multitool includes edit, erase and clear tools, while the **EditControlRegion** includes snapping and styling tools.

In the case of the Compact Toolbar, do not include the regions.

As of HTML5 Viewer 2.5, the **Erase** and **Clear** tools apply to both drawings and measurements.

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.


5606.38 m
Area: 66176361.93 m²; Perimeter: 35015.65 m
5606.40 m
4787.18 m
Area: 77980954.24 m²; Perimeter: 36695.73 m
5355.40 m

Measurements recalculated after the markup area is adjusted

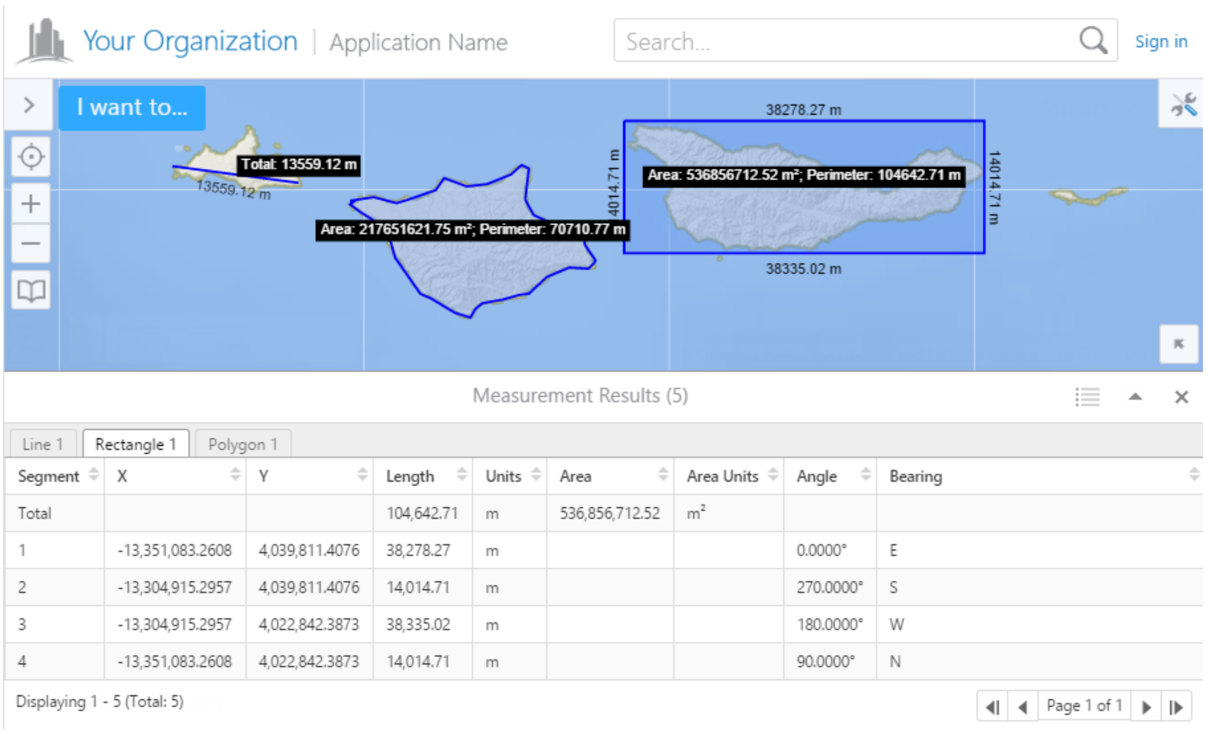
You can configure HTML5 viewers to show measurement data in the Results Table as well as on the map. This works in the Desktop and Tablet interfaces only—the Handheld interface does not have a Results Table.

Showing measurement data in the Results Table enables the user to export the data to a file. As well, the Results Table shows angle and bearing data, neither of which show on the map.

The Results Table provides an easy-to-read presentation of measurement data. In addition:

- The Results Table shows additional data that does not show on the map, namely, the angle and bearing.
- The Results Table has an Export function that enables the user to export the data to a file.

The Results Table opens automatically when the user completes the first measurement. Each additional measurement appears on a new tab.



Segment	X	Y	Length	Units	Area	Area Units	Angle	Bearing
Total			104,642.71	m	536,856,712.52	m²		
1	-13,351,083.2608	4,039,811.4076	38,278.27	m			0.0000°	E
2	-13,304,915.2957	4,039,811.4076	14,014.71	m			270.0000°	S
3	-13,304,915.2957	4,022,842.3873	38,335.02	m			180.0000°	W
4	-13,351,083.2608	4,022,842.3873	14,014.71	m			90.0000°	N

Displaying 1 - 5 (Total: 5)

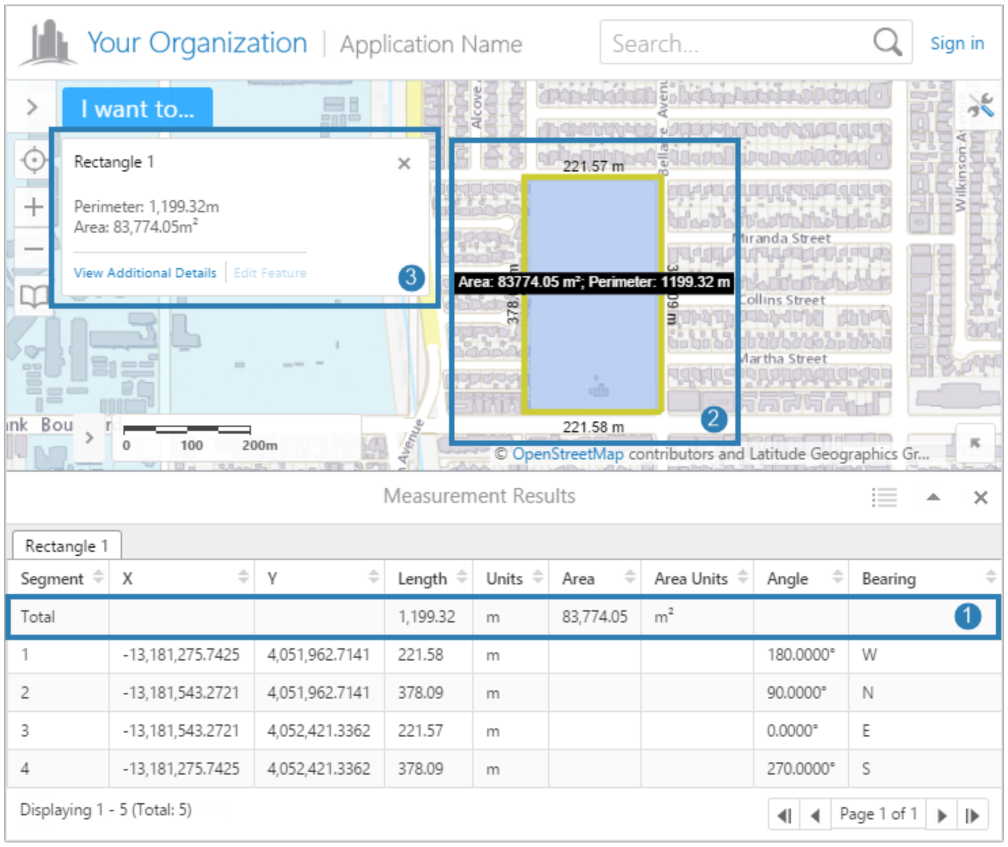
Measurement data in the Results Table

The first line in the table shows totals. If the measurement shape is made up of straight lines, the table contains additional lines of data—one line for each segment in the measurement.

The table is interactive. When the user clicks a row in the table, the map pans to the shape that the row represents, highlights the shape on the map, and opens a map tip containing the data from that row.

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 **180**.

The user can export the measurement data to a file by clicking the Export Results tool in the Panel Actions Menu, provided the Menu Module's `ResultsTableActions` has at least one export item configured. See **Menu Module** on page **197** for more information.



Rectangle 1

Perimeter: 1,199.32m
Area: 83,774.05m²

View Additional Details | Edit Feature

Measurement Results

Rectangle 1

Segment	X	Y	Length	Units	Area	Area Units	Angle	Bearing
Total			1,199.32	m	83,774.05	m²		
1	-13,181,275.7425	4,051,962.7141	221.58	m			180.0000°	W
2	-13,181,543.2721	4,051,962.7141	378.09	m			90.0000°	N
3	-13,181,543.2721	4,052,421.3362	221.57	m			0.0000°	E
4	-13,181,275.7425	4,052,421.3362	378.09	m			270.0000°	S

Displaying 1 - 5 (Total: 5)

Page 1 of 1

Click a row (1) to center the shape (2) and show the map tip (3)

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.

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 **48** 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`.
- **`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 list of map layers. The default name is **Drawings**.

    NOTE In order for the markup drawn by the Measurement Module to be editable by the Markup Module, the `markupLayerName` for both modules must be the same.

  - `lineColor`**:** The color of lines drawn on the map. The color can be a named string (**"red"**) or a hex string (**"#FF0000"**). The default color is **"#0000FF"** (blue).
  - `fillColor`**:** The fill color inside a polygon. The color can be a named string (**"green"**) or a hex string (**"#00FF00"**). The default color is **"#6495ED"** (cornflower blue).
  - `textColor`**:** The color of the measurement text markup. The color can be a named string (**"blue"**) or hex string (**"#0000FF"**). The default color is **"#000000"** (black) .
  - `textOffset`**:** The offset distance in pixels between the text labels and the geometry lines. The default is **5**.

    NOTE The offset value must be a number without an alphanumeric suffix.

  - `textSize`**:** The size of the measurement text. The default is **12px**.
  - `addMarkupToMapByDefault`**:** Defines whether or not markup is added to the map by default. The default is `true`.

See Also...

Configure Tool Behavior on page **91**

Configure the I Want To Menu on page **58**

Snapping Module on page **232**

MapTips Module on page **180**

Markup Module on page **184**

Results Module on page **213**

TabbedToolbar Module on page **235**

CompactToolbar Module on page **114**

About User Interface Text on page **48**

15.40 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.

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 **48** 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.
    - **`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.
    - **`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.
    - **`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.

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 **124**

FeatureDetails Module on page **129**

SimpleLayerList Module on page **232**

15.41 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 6.1+
- Android browser 4.0+

The following browsers support File Attachments on Mobile Phones:

- Android 4.0+
- Safari iOS 6.1+
- Chrome for Android (Android 4)

To see which other browsers support the HTML5 File Reader, see 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

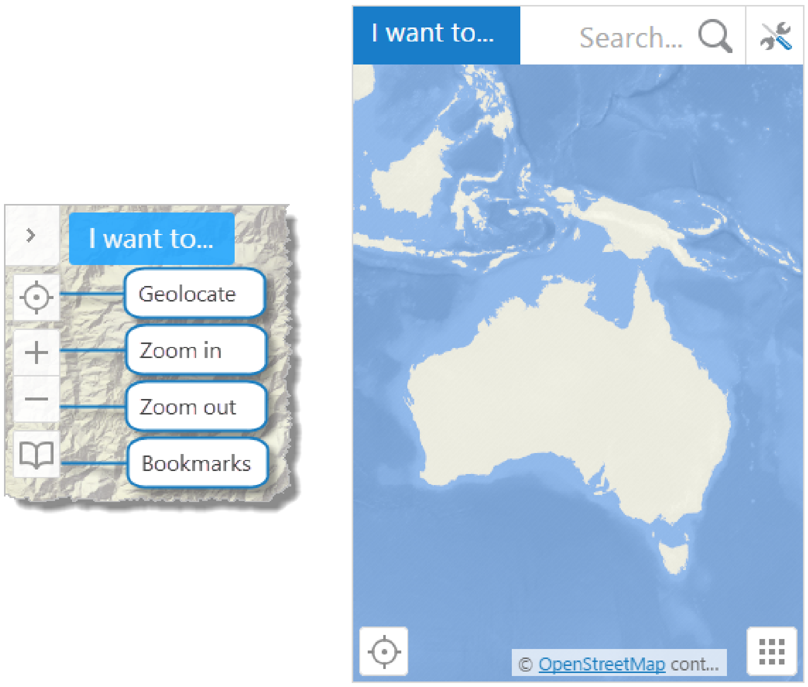
- **`AttachFileView`**: None

View Models

- **`AttachFileViewModel`**:
  - **`AttachFileViewId`**: The ID of the view in the NativeIntegration Module that displays file attachments.

15.42 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.

  NOTE 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 **52**.

- **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 **104**

Geolocate Module on page **144**

ZoomControl Module on page **250**

15.43 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 **67** for instructions.

The Offline Module allows the end user to run the viewer and interact with the map, including editing features, without being connected to a network.

The Offline Module 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

- **`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.
- **`ConnectionStatusIndicatorView`:** None
- **`ClearDataView`:** None

View Models

- **`ConnectionStatusIndicatorViewModel`:**
    - **`enabled`:** To enable the connection status indicator, set to `true`; otherwise set to `false`. The default is `false`. The menu is configured in the Menu Module.
- **`ClearDataViewModel`:** None
- (Desktop and Tablet interfaces only) **`OfflineMapDataMenuViewModel`:** None

See Also...

Editing Module on page **124**

FeatureLayer Module on page **137**

15.44 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...

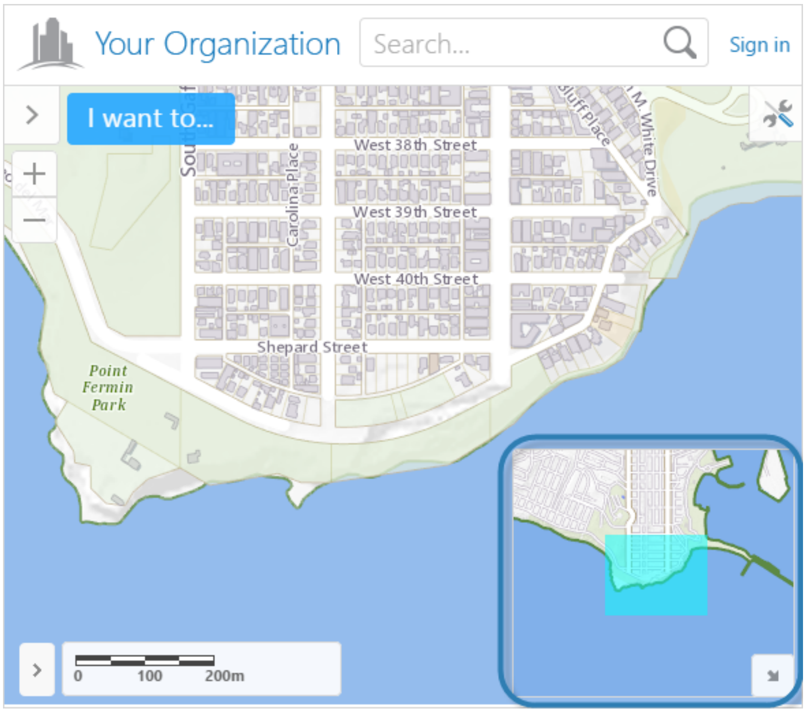
InsightIntegration Module on page **152**

15.45 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 **63** 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`.

    NOTE 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`.

15.46 Printing Module

The Printing Module implements simple printing of the map image.

In order for the user to be able to print the map, you must create at least one print template using the Geocortex Report Designer. Refer to the Report Designer help system for information.

After you have created the print template, you must add it to your site. The user can print the map using any print template that you add to the site. 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. Refer to the *Geocortex Essentials Administrator Guide* for information on adding and editing print templates.

The factory HTML5 Viewer has an item in the I Want To menu to print the map, as well as a tool in the toolbar.

If you want print files to open automatically for the user, select the Automatically Open URLs check box on the viewer's Application page in Manager. If the user's browser blocks pop-ups, the user may have to give permission for the file to open. If you clear the Automatically Open URLs check box, users must click a button to download the file, and then open the file.

Version 2.0 of the HTML5 Viewer does not support large format printing.

If the site does not have any print templates, you might want to remove the print options from the I Want To menu and the toolbar.

Configuration Properties

Module

None

Views

- **`PrintingView`**: None

View Models

- **`PrintingViewModel`**: None

15.47 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

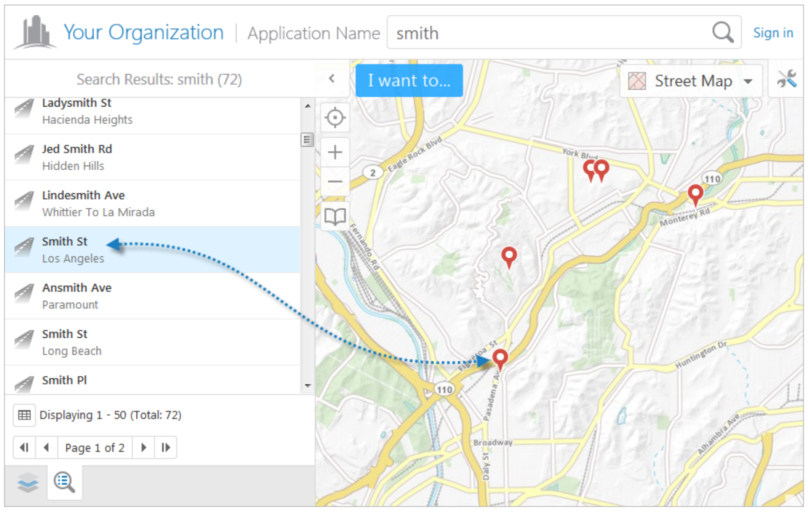
The Prompt Module does not have any views.

View Models

The Prompt Module does not have any view models.

15.48 Pushpins Module

The Pushpins Module is responsible for placing a pushpin on the map for each feature on the current page of the search results. The pushpin is placed in the center of each feature, with a specified offset. Clicking the pushpin displays a map tip, if map tips are configured.



Pushpins are not currently supported in the Handheld interface. Adding pushpin configuration to the Handheld configuration file has no effect.

If you add markup to a map that has pushpins displayed, the markup appears behind the pushpins by design.

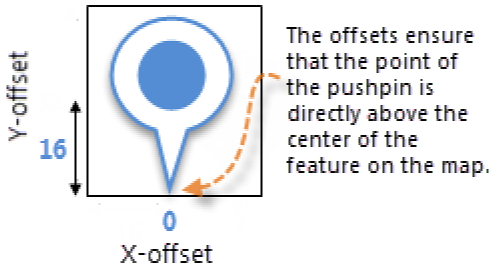
Pushpins are supported with geocoders and workflows.

Configuration Properties

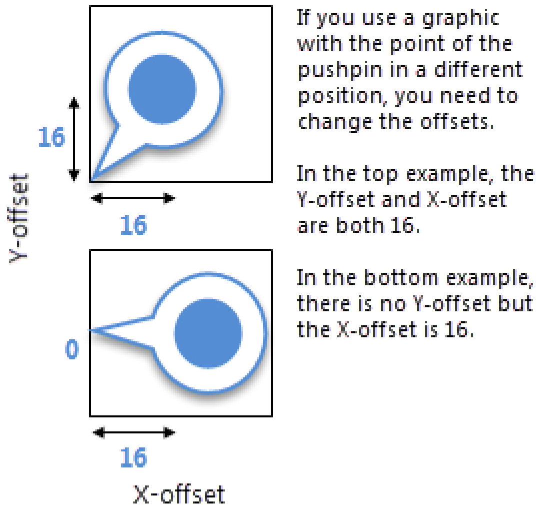
Module

- **`pushpinsEnabled`:** Set to `true` if you want pushpins to appear for each feature of the search results; otherwise, set to `false`. The default is `true`.
- **`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.



The offsets ensure that the point of the pushpin is directly above the center of the feature on the map.



If you use a graphic with the point of the pushpin in a different position, you need to change the offsets.

In the top example, the Y-offset and X-offset are both 16.

In the bottom example, there is no Y-offset but the X-offset is 16.

- **`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`.
  - **`PushpinMouseLeftButtonDownBehavior:`** 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 advanced 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 **180**

15.49 QueryBuilder Module

The QueryBuilder Module implements the Query Builder and Filter Builder.

Query Builder

The Query Builder enables end users to construct queries that search for specific features on a layer. The results of a query list the features in the specified layer that match the query conditions. In addition, each feature in the query results is marked on the map with a pushpin 📍. The features do not have to be visible at the current scale for the pushpins to show.

For example, a user could construct a query that searches the **Hospitals** layer for medical facilities with **Clinic** in the name that are located in the city of **Pleasantville**. Clicking **Search** lists Pleasantville's clinics and marks each clinic on the map with a pushpin. In the factory configuration, the user can click a clinic in the query results, or click the clinic on the map, to see additional details about the clinic.

WMS layers that are not associated with a WFS do not support query operations.

Filter Builder

The Filter Builder enables end users to filter the features that show on the map. Filtering also hides query pushpins, if there are any. Filtering does not affect which features are listed in search or query results. The user can **Clear** a filter to show the features and pushpins that were hidden by filtering. Filtering is the same as applying a definition expression in ArcMap.

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. If any of the hidden features are marked by a search or query pushpin, the pushpins are hidden also. Clicking **Clear** re-displays the hidden features and pushpins on the map.

15.49.1 Configure the Query Builder and Filter Builder

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 "How to Configure 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 **79**.

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 **79**.

    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

None

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.

Your data provider's date format can contain any of the .NET date and time format specifiers that Essentials supports. 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`.
Essentials supports the following .NET date and time format specifiers:

`d`	`fffffff`	`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.

- **`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.
    Your data provider's date format can contain any of the .NET date and time format specifiers that Essentials supports. 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`.

Essentials supports the following .NET date and time format specifiers:

`d`	`fffffff`	`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`	

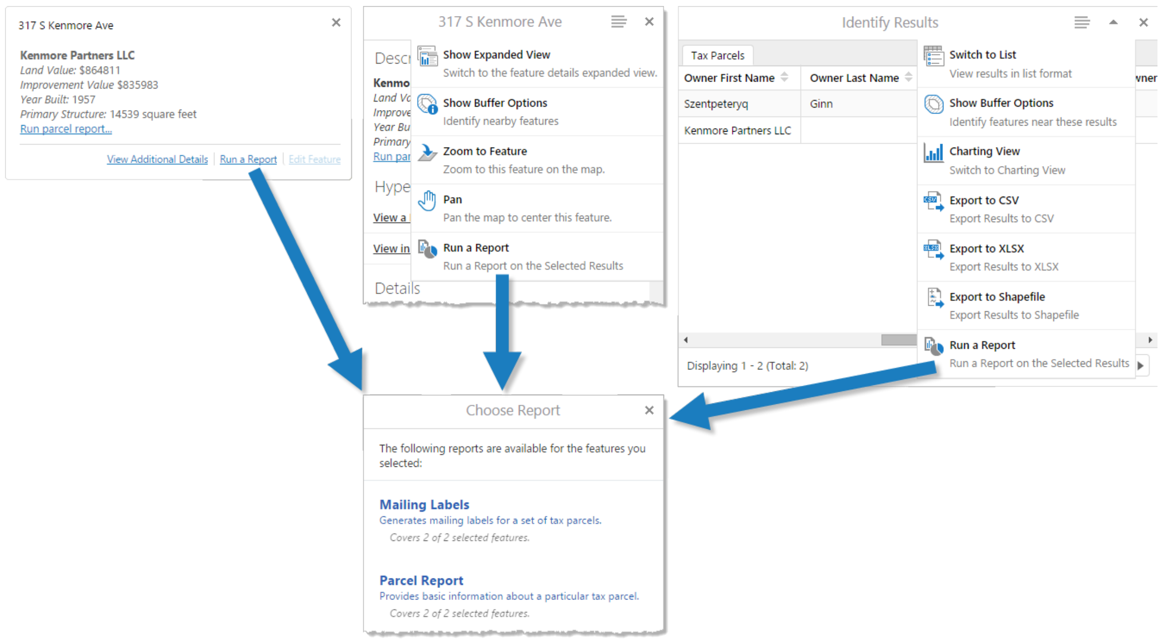
- **`textComparisonQueryFormat`**: The format of a query that compares the value of a text field to the input value. For example, the format for text comparisons in the ANSI SQL default data provider is `LOWER({0}) LIKE LOWER({1})`.
- **`numberToTextComparisonQueryFormat`**: The format of a query that 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.

View Models

- **`SimpleQueryBuilderViewModel`**:
  None
- **`SimpleFilterBuilderViewModel`**:
  None

15.50 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*.



317 S Kenmore Ave

Kenmore Partners LLC
Land Value: $864811
Improvement Value $835983
Year Built: 1957
Primary Structure: 14539 square feet
Run parcel report...

Owner First Name	Owner Last Name
Szentpeteryq	Ginn
Kenmore Partners LLC	

Choose Report

The following reports are available for the features you selected:

Mailing Labels
Generates mailing labels for a set of tax parcels.
Covers 2 of 2 selected features.

Parcel Report
Provides basic information about a particular tax parcel.
Covers 2 of 2 selected features.

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

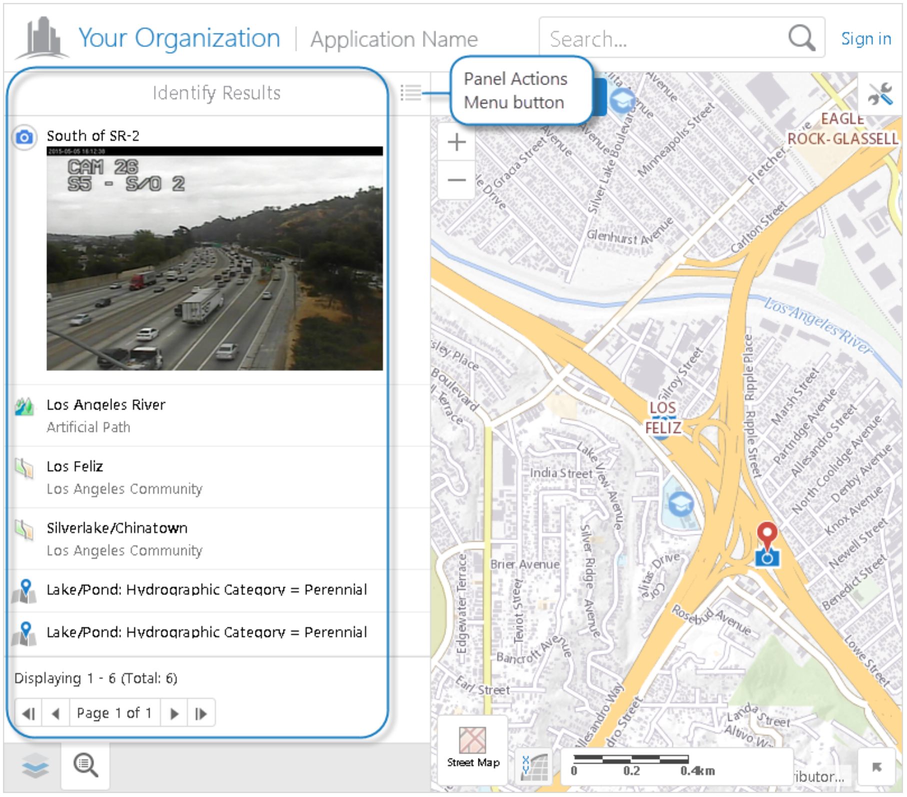
- `ListReportsViewModel`: None
- `RunReportViewModel`: None

15.51 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: Compact View and Expanded View. In the Compact View, results are displayed as a list in the sidebar. In the Expanded View, results 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, results are displayed in the Compact View. The Handheld interface does not support the Expanded View. The Compact View can be activated via the `ShowResultsList` command, while the Expanded View can be activated via the `ShowResultsTable` command.

The Compact View displays results as a list in the sidebar, as illustrated below.



South of SR-2

Los Angeles River
Artificial Path

Los Feliz
Los Angeles Community

Silverlake/Chinatown
Los Angeles Community

Lake/Pond: Hydrographic Category = Perennial

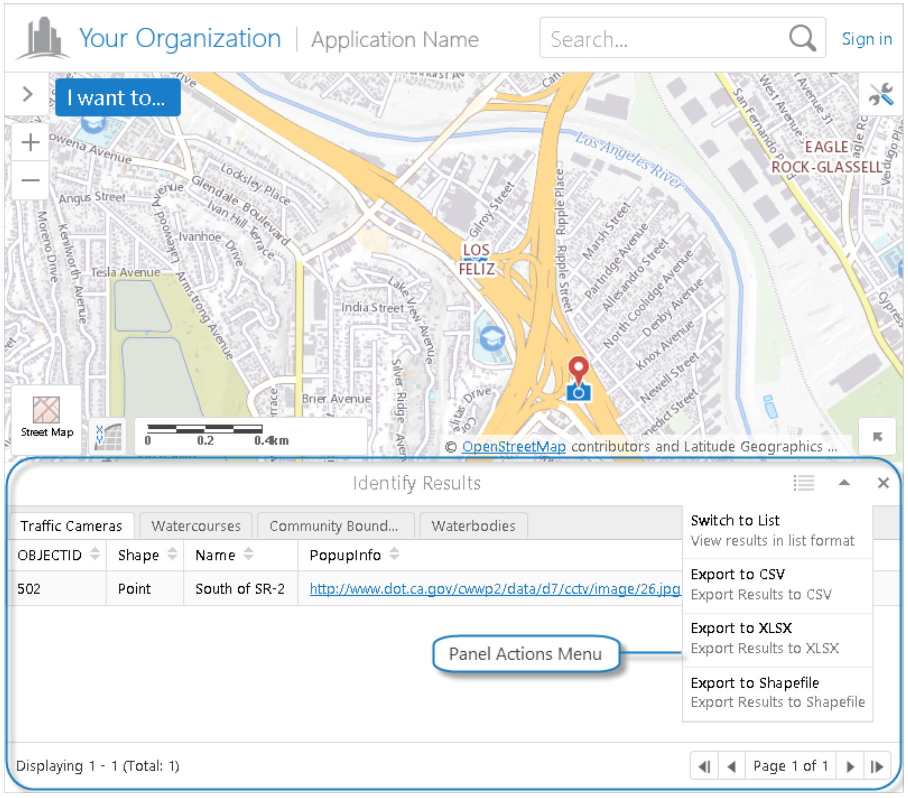
Lake/Pond: Hydrographic Category = Perennial

Displaying 1 - 6 (Total: 6)

Page 1 of 1

Example of results list in the Compact View

The Expanded View displays results in a table within separate tabs for each category, as illustrated below.



Traffic Cameras	Watercourses	Community Bound...	Waterbodies

OBJECTID	Shape	Name	PopupInfo
502	Point	South of SR-2	http://www.dot.ca.gov/cwwp2/data/d7/cctv/image/26.jpg

Example of results table in the Expanded View

Furthermore, the Panel Actions Menu that contains options for exporting the results' attributes to a `.csv` file, an `.xlsx` file, or a shapefile (`.shp` file). 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 Builder, measurement, or map tip operation, possibly run from a workflow. You configure the mapping between the source and target (the module that displays the results) of result sets using the `resultMappings` element.

Configuration features for the Results Module include an `eventMappings` element that enables events to be mapped to an independent array of command names. The command names in turn, take a feature as an argument, and will be executed when a specific event is fired.

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 a set of three commands: `AddPushpins`, `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 a single 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 **189**.
    - `Workflow:` An array of commands to execute when the feature set collection was generated from a search, identify, map tip, or query operation that was executed in a workflow. The default is a set of three commands: `AddPushpins`, `ShowResultsList` and `SetCollectionOfInterest`.
    - `Search:` An array of commands to execute when the feature set collection was generated from a search operation. The default is a set of three commands: `AddPushpins`, `ShowResultsList` and `SetCollectionOfInterest`.
    - `QueryBuilder`**:** An array of commands to execute when the feature set collection was generated from a QueryBuilder operation. The default is a set of three commands: `AddPushpins`, `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 a single 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 a single command: `ShowFeatureDetails`.
    - `ResultsTableFeatureClickedBehavior:` A behavior that runs an array of commands when a feature in the Results Table is clicked. By default, this includes a single command: `ShowMapTip`.
    - `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 a single command: `ShowMapTip`.

  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 advanced 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**.

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`.
  - `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.
- `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
- `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, `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 `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`, **`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...

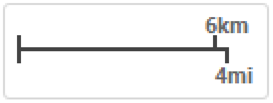
Menu Module on page **197**

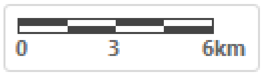
Measurement Module on page **189**

15.52 Scalebar Module

This module can be configured using Manager. For instructions, see **Configure Map Widgets** on page **63**.

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)

Configuration Properties

Module

None

Views

- `ScalebarView`:

  To hide the scale bar, set the view's `visible` property to `false`; otherwise, leave it as `true`.

  - `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.

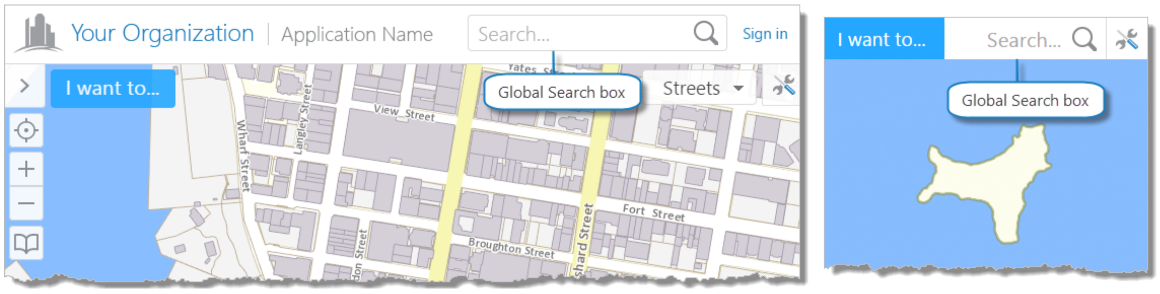
  - `showBackground`**:** To display a background for the scale bar, set to `true`; otherwise, set to `false`. The default is `true`.

View Models

- `ScalebarViewModel`: None

15.53 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 **213**.

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 **213**

15.54 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.

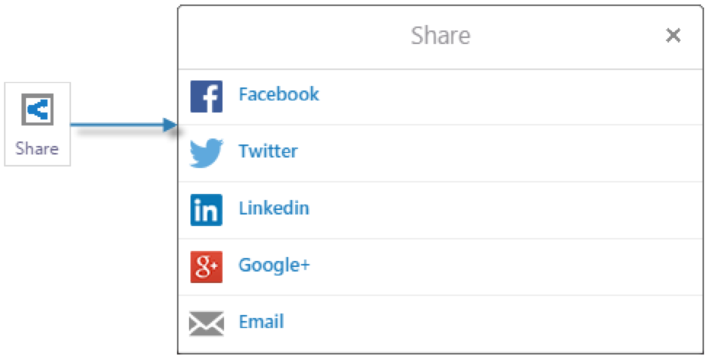
Sharing **options** are the application types (for example, email) or the specific applications (for example, Facebook) that the user can share links with. Options are configured using the Share Module's `shareOptions` property. By default, `shareOptions` are configured for Facebook, Twitter, LinkedIn, 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

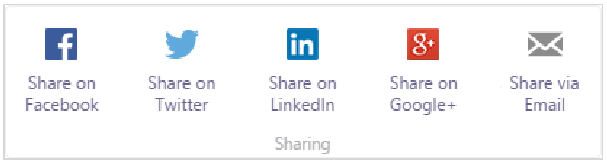
To list the sharing options, use the `ShowShareView` command, which displays the list of sharing options (the Share Module's `ShareView` view). The HTML5 Viewer has a Share tool that you can add to the toolbar to run the `ShowShareView` command.



The Share tool (left) lists the sharing options when clicked

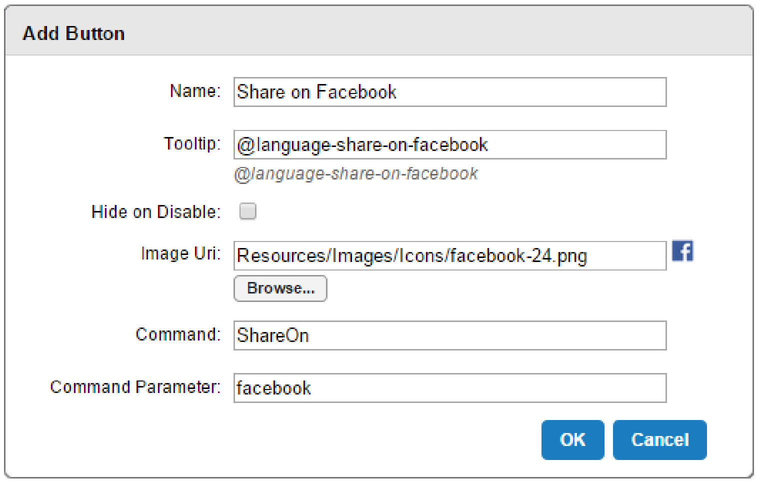
Provide Separate Options

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: Share on Facebook

Tooltip: @language-share-on-facebook
@language-share-on-facebook

Hide on Disable: ☐

Image Uri: Resources/Images/Icons/facebook-24.png

Command: ShareOn

Command Parameter: facebook

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 **224**.
  - **`iconUri`:** The relative URI 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 **224**
>
> **Geocortex SDK for HTML5 API Reference** on page **269**

15.55 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 `configBase` parameter to point to the viewer's configuration files. When the SharingLink Module assembles a URL to share, it copies the `configBase` 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 `configBase`, `extent`, and `runWorkflow` parameters. Under the default configuration, the SharingLink Module copies `configBase` 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 **36**.

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 paramters is given in **URL Parameters Reference** on page **36**.

- **`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 **222**

Geocortex SDK for HTML5 API Reference on page **269**

URL Parameters Reference on page **36**

15.56 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`.
- **`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`.
- **`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`.
  - **`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`.
    - `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.

15.57 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 **170**

15.58 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

15.59 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`. The **Snappable Layers** panel, which is opened by the **Select Snapping Layers** button, is configured in the LayerSelector Module.

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`.

Views

The Snapping Module does not have any views.

View Models

The Snapping Module does not have any view models.

See Also...

TabbedToolbar Module on page **235**

CompactToolbar Module on page **114**

Measurement Module on page **189**

Markup Module on page **184**

Editing Module on page **124**

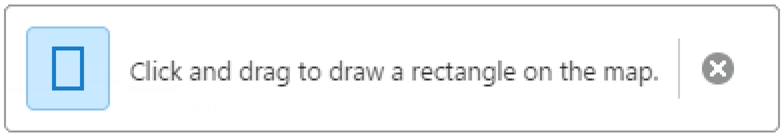
Identify Module on page **149**

LayerSelector Module on page **171**

15.60 Status Module

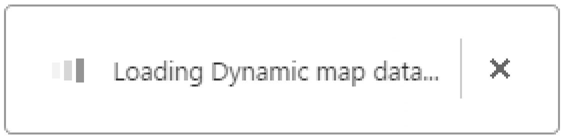
The Status Module is a general-purpose module that can be used by any module to show a status message. There are generally two status types:

- 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.



Click and drag to draw 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.



Loading Dynamic map data...

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.

View Models

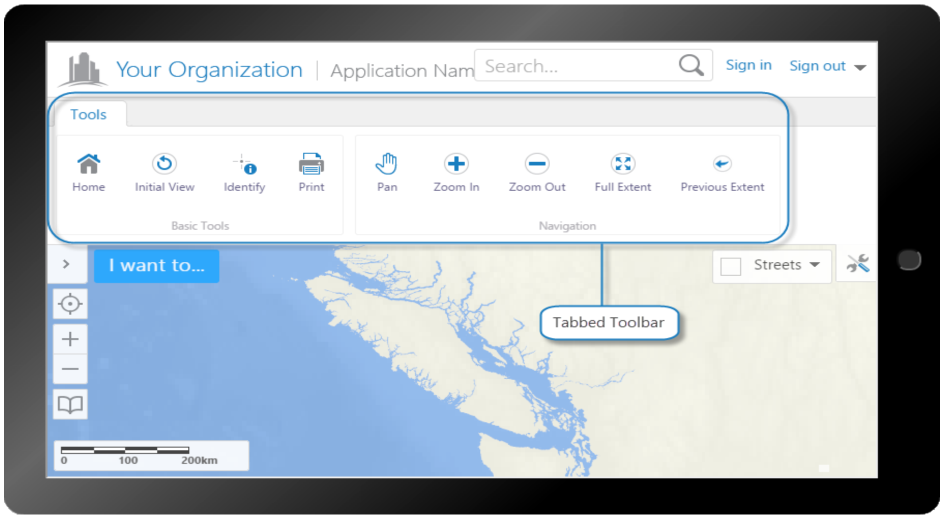
The Status Module does not have any view models.

15.61 TabbedToolbar Module

The toolbar can be configured using Manager. See **Configure the Toolbar** on page **79** 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.

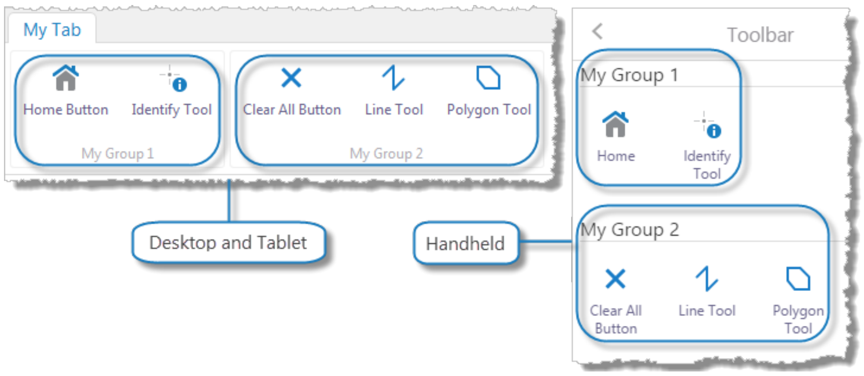


The Tablet interface, showing the Tabbed Toolbar

Tabbed Toolbar

The Tabbed Toolbar is made up of tabs, groups, buttons, tools, and 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.

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 **296**.

  - **`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 **269**.
- **`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 **269**.
- **`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 **48** 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 **269**.
      - `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 **269**.
      - `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 **269**.
- `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
- **`NavButtonView`:** (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
- **`NavButtonViewModel`:** (Handheld interface only) None

See Also...

About User Interface Text on page **48**

CompactToolbar Module on page **114**

15.62 Toolbar Module

As of Geocortex Viewer for HTML5 2.4, the Toolbar Module no longer exists and has been replaced by the TabbedToolbar Module.

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.

See Also...

TabbedToolbar Module on page **235**

CompactToolbar Module on page **114**

15.63 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.

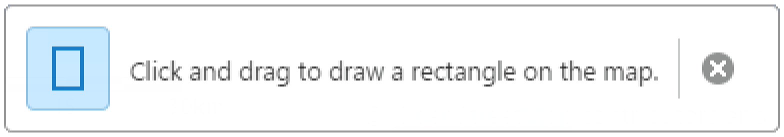
Configuration Properties

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.

  
  Click and drag to draw a rectangle 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 **48** for more information on using text keys.

  - **`command`:** The command that the tool runs.

    For a list of commands, see 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 **124**

Identify Module on page **149**

IWantToMenu Module on page **168**

About User Interface Text on page **48**

15.64 User Module

The User Module implements user sign-in and sign-out to sites that are secured using Essentials Security.

In the Desktop and Tablet interfaces, users initiate sign-in and sign-out by clicking the sign-in or sign-out hyperlink. You can enable and disable the hyperlinks in Manager.

The Handheld interface does not have sign-in and sign-out hyperlinks. To allow users to sign in and sign out, you must make the `SignIn` and `SignOut` commands available. Typically, you would add Sign In and Sign Out items to the I Want To menu.

The Desktop and Tablet interfaces have two views and two view models. `SignInView` and `SignInViewModel` control sign-in. `UserInfoView` and `UserInfoViewModel` control sign-out. The Handheld interface does not have any views or view models.

By default, the sign-in and sign-out hyperlinks are in the viewer's `BannerContentRegion`.

For information about Essentials Security, refer to the *Geocortex Essentials Administrator Guide*.

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`:** A valid HTML color to use for the sign-in hyperlink's text, for example, **red** or **#FF0000**. By default, the hyperlink is blue. If the default color does not show up well against the background color, change `linkColor` to a color that contrasts better.
- **`UserInfoViewModel`:**
    - **`linkColor`:** A valid HTML color to use for the sign-out hyperlink's text, for example, **white** or **#FFFFFF**. By default, the hyperlink is blue. If the default color does not show up well against the background color, change `linkColor` to a color that contrasts better.
    - **`textColor`:** A valid HTML color to use for the text in the panel that opens when the user clicks the down arrow icon ▼ beside the sign-out hyperlink. Configure `textColor` and `backgroundColor` so the panel's text is easy to read.
    - **`backgroundColor`:** A valid HTML color to use for the background of the panel that opens when the user clicks the down arrow icon ▼ beside the sign-out hyperlink. Configure `textColor` and `backgroundColor` so the panel's text is easy to read.

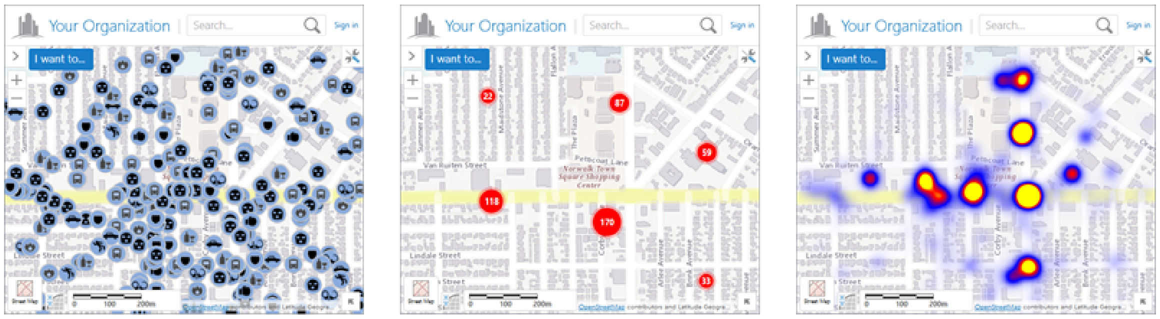
See also...

[**Geocortex SDK for HTML5 API Reference** on page **269**]

[**Configure the I Want To Menu** on page **58**]

15.65 Visualization Module

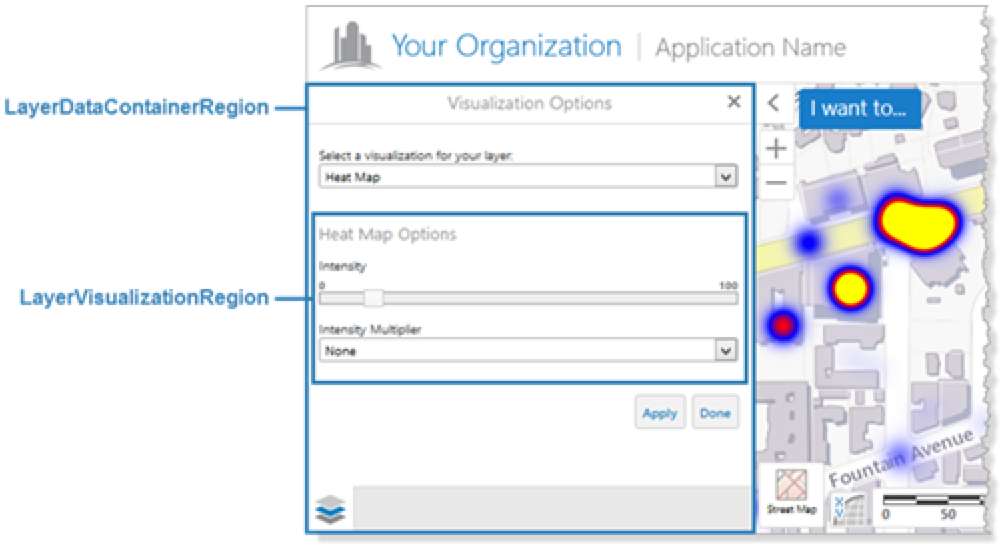
The Visualization Module is a container that hosts visualization providers. Visualization providers provide different ways of representing features on the map. By default, there are two visualization providers, one for clustering and one for heat maps. The visualization providers themselves are implemented by their respective modules, the ClusterLayers Module and the HeatMaps Module.



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. `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 two providers, one for heat maps and 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...

ClusterLayers Module on page **113**

HeatMaps Module on page **147**

15.66 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

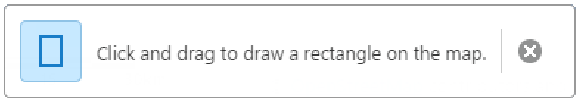
Starting in version 2.4 of the HTML5 Viewer, the `startupWorkflows` module property is deprecated. Startup workflows are now configured in the site. For information, see "Configure a Workflow to Run on Startup" in the *Geocortex Essentials Administrator Guide*. Startup workflows that were configured 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 **48** 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`.

  
  Click and drag to draw a rectangle on the map.

  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 **48**

Status Module on page **234**

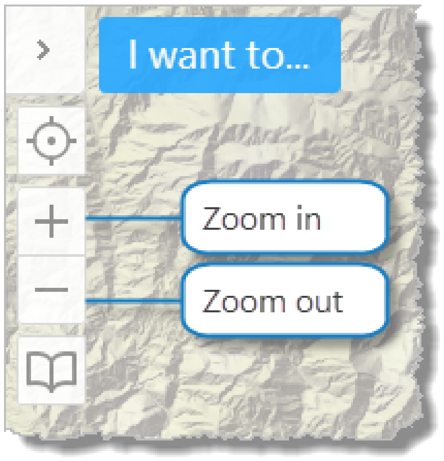
Tools Module on page **244**

15.67 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 **199**

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:

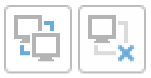
1. Find the command in the Geocortex SDK for HTML5 API Reference.
2. Look in the command's **Description** column.
   Commands that cannot be used in workflows have a note in the description column. If there is no such note, then you can safely use the command in workflows. Some commands have restrictions on how they are used—these restrictions are also described.

17 Offline Mode

17.1 About Offline Mode

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.

Users working with an offline-enabled viewer can manually switch between Offline and Online modes by clicking or tapping the Online/Offline icon in the bottom right corner of the screen.



Online/Offline icons

From version 2.0 of the Geocortex Viewer for HTML5, Offline mode requires the Geocortex Mobile App Framework.
To download and install the Geocortex Mobile App Framework, please contact Latitude Support.
For more information, please see the *Geocortex Mobile App Framework Administrator Guide*.

As of Geocortex Mobile App Framework 1.3, all Essentials security options work offline.

17.1.1 Configure Offline Mode

You configure Offline mode in Manager and the Geocortex Mobile App Framework. For information on configuration please see the following documentation:

- *Geocortex Mobile App Framework Administrator Guide*.
- *Geocortex Essentials Administrator Guide*.

17.1.2 Offline Editing and Syncing

Sites that are taken offline usually include editable feature layers, or feature layers that have navigation data. The HTML5 Viewer supports offline editing when using these layers, including creating, modifying, and removing features, as well as editing related features.

When you make edits to a map while offline, the edits are stored in a log in chronological order. When you then go online again, the logged edits are pushed to the server using the Sync tool. Before you synchronize, you can review the edits, re-edit them, or undo them.

During synchronization, some edits may fail due to validation rules that exist on the server. Edits may also fail when feature services or their underlying database schema have changed.

When edits fail to synchronize, they are highlighted on the Manage Offline Data panel. You then need to repair or undo the failed edits. When all the edits have been uploaded to the server or discarded, the synchronization process refreshes the cached data by downloading any changes that have been made on the server.

18 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 2.5 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.

18.1 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.

18.2 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.

NOTE 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.

NOTE 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...
Navigate forward through the page's components	TAB *
Navigate backward through the page's components	SHIFT + TAB
Select or activate the current UI component	ENTER
Select checkbox	SPACE BAR
Pan the map (if selected)	Arrow keys **
Move slider	Arrow keys
Jump slider	PAGE UP; PAGE DOWN
Jump slider to the start or end	HOME; END

***** 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.

****** 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.

HTML5 Viewer Keyboard Shortcuts for Accessibility

To do this...	Press...
Add a vertex to the shape that you are creating	ENTER
Movevertex horizontally or vertically	Arrow keys
Movevertex diagonally	PAGE UP; PAGE DOWN; HOME; END
Move the selected shape horizontally or vertically	Arrow keys
Move the selected shape diagonally	PAGE UP; PAGE DOWN; HOME; END
Enlarge the selected shape uniformly	S

To do this...	Press...
Reduce the selected shape uniformly	SHIFT + S
Rotate the selected shape to the right	R
Rotate the selected shape to the left	SHIFT + R
Complete the shape	ENTER, ENTER (press ENTER twice)
Enter vertex editing mode	V
Select the next vertex of the current shape (in vertex editing mode)	V
Select the previous vertex of the current shape (in vertex editing mode)	SHIFT + V
Delete vertex	D
Exit vertex editing mode	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

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 may 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 may times as needed to navigate to the **Draw** multitool.
2. Press **ENTER** to open the **Draw** multitool.
3. Press **TAB** as may 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 **97**

19 Translation

19.1 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.

19.2 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`

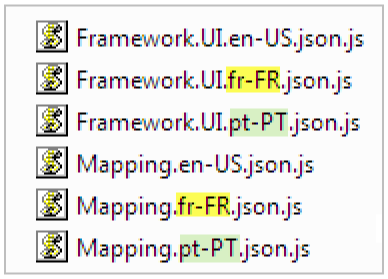
To translate the language files:

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.

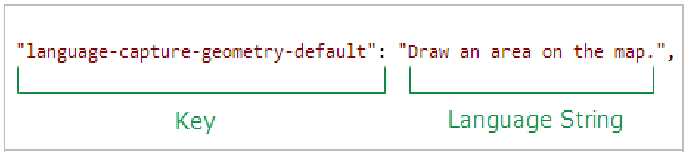
3. Rename the language files with the language code of the language you are translating the viewer to.



```
Framework.UI.en-US.json.js
Framework.UI.fr-FR.json.js
Framework.UI.pt-PT.json.js
Mapping.en-US.json.js
Mapping.fr-FR.json.js
Mapping.pt-PT.json.js
```

Language files copied and renamed for French and Portuguese

4. Open each language file in a text editor and translate each language string.



```
"language-capture-geometry-default": "Draw an area on the map.",
```

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.

5. Open the three configuration files, `Desktop.json.js`, `Handheld.json.js`, and `Tablet.json.js`, in a text editor such as Notepad.

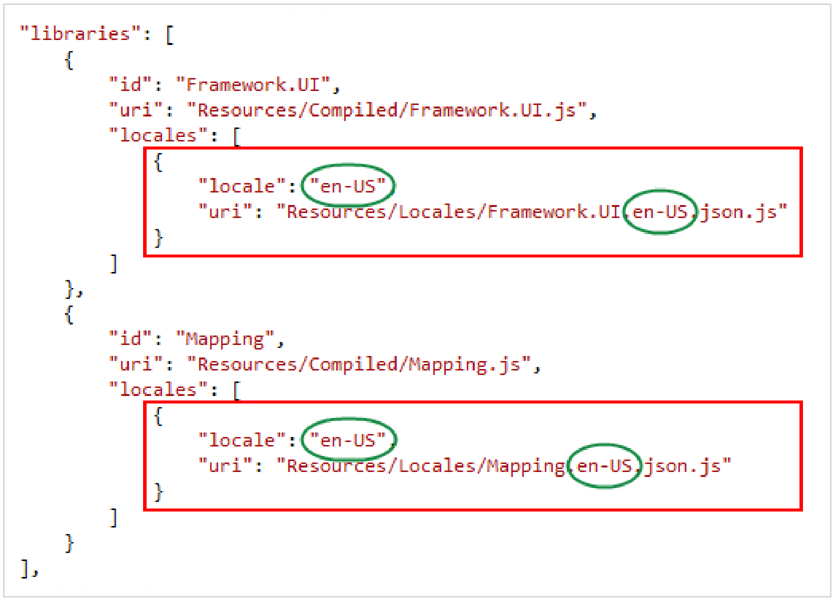
   The configuration files are in the viewer's virtual directory. The default location is:

   ```
   C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\
   Default\RESTElements\Sites\<site>\Viewers\<viewer>\
   VirtualDirectory\Resources\Config\Default
   ```

   For more information, see**File Locations** on page **292**.

6. 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.

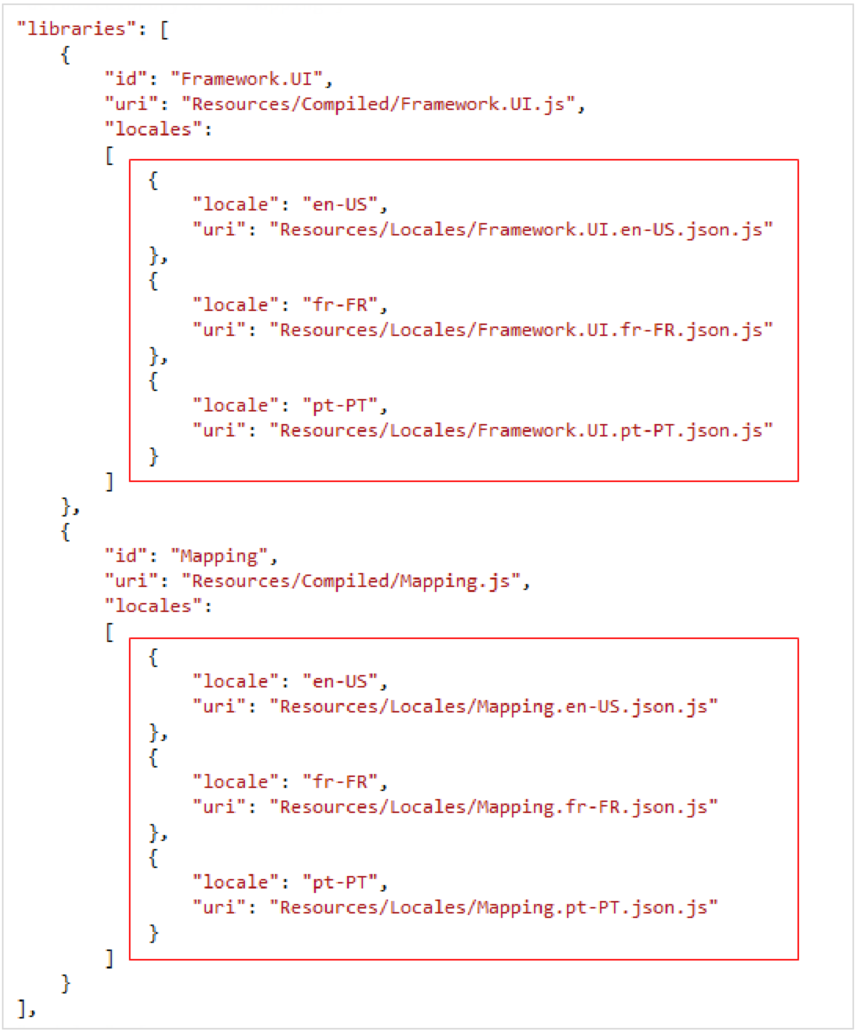


```
"libraries": [
    {
        "id": "Framework.UI",
        "uri": "Resources/Compiled/Framework.UI.js",
        "locales": [
            {
                "locale": "en-US",
                "uri": "Resources/Locales/Framework.UI.en-US.json.js"
            }
        ]
    },
    {
        "id": "Mapping",
        "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.

8. 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

9. Save the configuration files and test the viewers.

20 Custom Development

20.1 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/3.14/dijit/tree/TreeStoreModel.js` and save it to `Resources/Scripts/dijit/tree/TreeStoreModel.js` within your HTML5 Viewer folder.

20.2 Key Concepts

20.2.1 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.

20.2.2 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**.

20.2.3 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.

20.2.4 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.

(Developers with Silverlight/WPF or Flex experience will be familiar with the style of data binding used in the Geocortex HTML5 framework.)

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.

20.2.5 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. 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.

In the Silverlight world, a library can be thought of as similar to a XAP 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 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 compiler. Java must be on your PATH environment variable in order to run the SDK build scripts.

See also...

[**Resource Compiler Tool** on page **272**]

20.2.6 UI Composition – Regions and Views

The Geocortex Viewer for HTML5 uses a style of user interface (UI) composition that should be familiar to those who have worked with the Geocortex Viewer for Silverlight and/or the Prism framework.

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 **288**.

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 than 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.

20.2.7 UI Composition – 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 **288**.

20.2.8 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 **269**

20.2.9 States

Geocortex Mobile App Framework 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 **296**.

See also...

Configure the Toolbar on page **79**

CompactToolbar Module on page **114**

TabbedToolbar Module on page **235**

State Reference on page **296**

20.3 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.

20.3.1 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 **269**

20.4 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.

To access the Geocortex SDK for HTML5 API Reference:

1. Download 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.

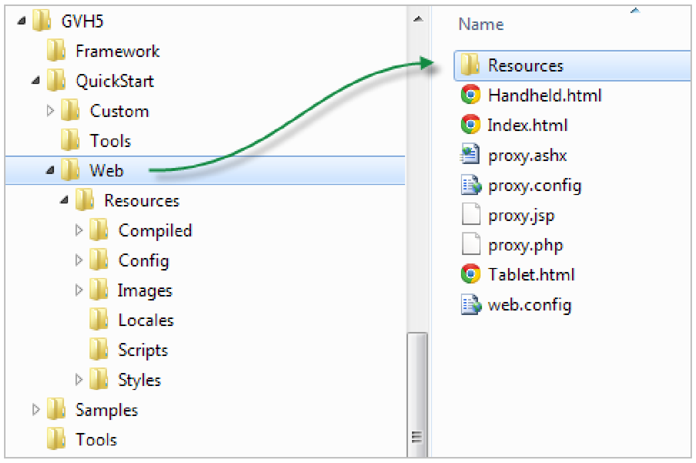
20.5 Project Layout

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.

20.6 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.

20.7 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.

20.8 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/.

20.9 The Resource Compiler

20.9.1 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).

20.9.2 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.

20.10 Data Binding

20.10.1 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.

20.10.2 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 **274**.
  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.

20.10.3 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`.

20.10.4 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 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.

20.10.5 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.

20.10.6 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 some 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>
```

20.10.7 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.

20.10.8 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 `CollectionChangedArgs` 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 `CollectionChangedArgs` items.

The use of `ObservableCollection` facilitates one of the most powerful types of data binding: collection binding.

20.10.9 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`.

20.11 HTML5 Viewer Embedding

20.11.1 About HTML5 Viewer Embedding

20.11.1.1 Architecture

Before embedding the Geocortex Viewer for HTML5 (GVH), 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`.

20.11.1.2 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:

- **`baseUrl?`:** 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.

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.

20.11.1.3 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.

20.11.1.4 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`

20.11.2 Embed an HTML5 Viewer

20.11.2.1 HTML Head Section

In your HTML, we recommend you configure the following optional metadata tags within your `<head>` tag:

- Declare some basic `<meta>` tags:

```
<meta charset="utf-8"/>
<meta name="viewport" content="width=device-width,user-scalable=no,initial-
scale=1"/>
<meta http-equiv="X-UA-Compatible" content="IE=edge"/>
```

- Declare your own website title:

```
<title>Geocortex Essentials HTML5 Viewer</title>
```

- Declare your own website icon:

  ```
  <link rel="shortcut icon" href="favicon.ico"/>
  ```

20.11.2.2 Basic Example

The most basic way to embed a Geocortex Viewer for HTML5 (GVH) within your HTML, to add the following `<script>` tags just before the end of your `</body>` tag:

```
<script src="Resources/Compiled/loader.js"></script>
<script>
    new geocortex.essentialsHtmlViewer.ViewerLoader().loadAndInitialize();
</script>
```

The first `<script>` tag loads the necessary `loader.js` script. The second `<script>` tag creates a new default instance of a `ViewerApplication` object. All GVH dependencies and resources will be loaded.

20.11.2.3 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>
    new geocortex.essentialsHtmlViewer.ViewerLoader().loadAndInitialize({
            hostElement: document.body,
            configBase: "Resources/Config/Default/"
            onLoaded: function(viewer, viewerLoader {
                console.log("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 `onLoaded` callback function when the `ViewerLoader`'s `ResourceSet` has finished loading and the `ViewerApplication` has been initialized. In the above example, the callback function outputs the new viewer to the console.

20.11.2.4 Available Viewer Options

To customize the viewer created, the `loadAndInitialize()` method accepts a `ViewerInitializationOptions` parameter with the following optional properties:

- **`hostElement?`** (`HTMLElement`): The DOM element to host the application. The default is `document.body`.
- **`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 }`): Aliases to the full `viewerConfigUri` paths. An alias can be specified by the `viewer` query string 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/"`.
- **`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.

20.11.2.5 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">
           <img class="splash-image" src="Resources/Images/splash-logo.png" alt=""/>
           <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>
   ```

20.11.2.6 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 **base maps** and **operational maps**. Base maps supply background and contextual information in layers. Base maps 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 base maps, 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.

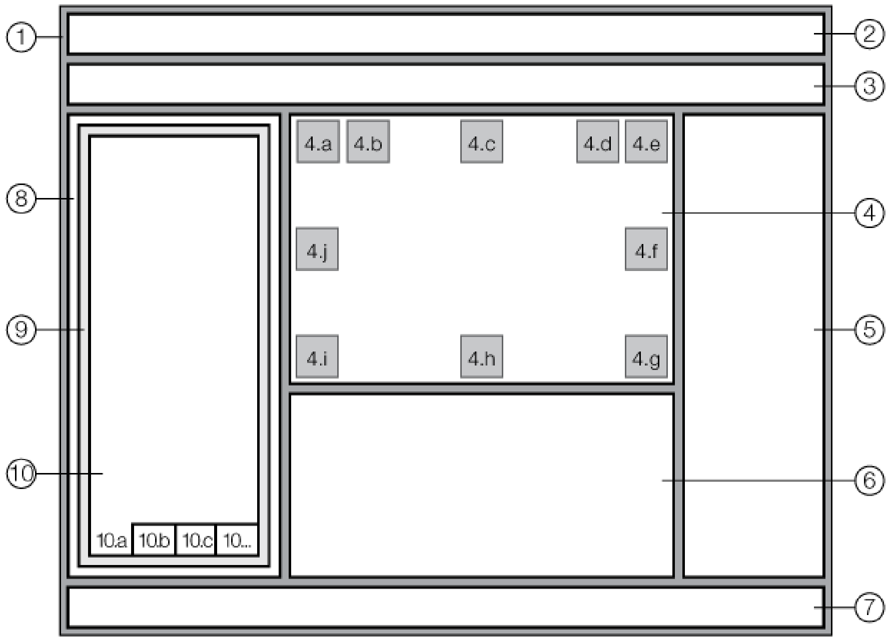
Appendix B: Regions

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 Base Map 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`**

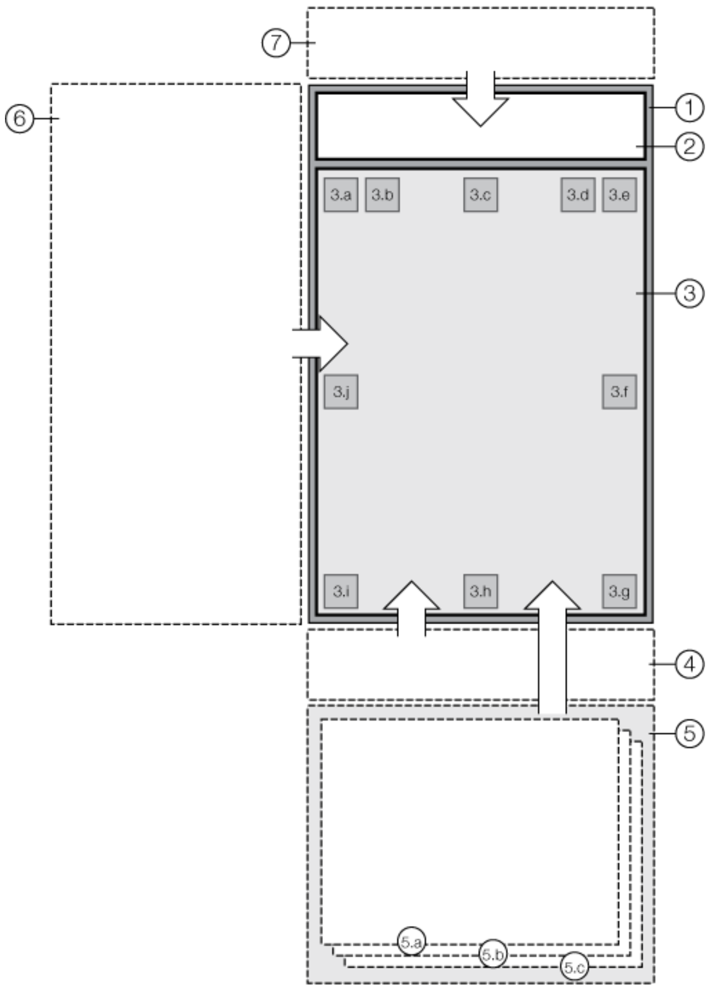
5. **`RightPanelRegion`**
6. **`ResultsRegion`**: Holds the Results Table and other results-related views, like feature details. `ResultsRegion` is in the `BottomPanelRegion` (not shown in the diagram).
7. **`FooterRegion`**: Holds the application's footer. The `FooterRegion` has two regions on top of 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.
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`.



1. `ApplicationRegion`
2. `HeaderRegion`
3. `ScreenRegion`
    a. `TopLeftMapRegion`
    b. `NavigationMapRegion`
    c. `TopCenterMapRegion`
    d. `BasemapMapRegion`
    e. `TopRightMapRegion`
    f. `MiddleRightMapRegion`
    g. `BottomRightMapRegion`
    h. `BottomCenterMapRegion`
    i. `BottomLeftMapRegion`
    j. `MiddleLeftMapRegion`
4. `SmallNavigationRegion`
5. `BottomPanelRegion`
    a. `LayerDataContainerRegion`
    b. `ResultsViewContainerRegion`
    c. `GeneralViewContainerRegion`
6. `LeftPanelRegion`
7. `TopRightShellRegion`

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 GeneralViewContainerRegion. 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.
- `GeneralViewContainerRegion`**:** 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`
Folder for a particular site		`{GE_SITES}\[site]`
Site configuration file		`{GE_SITES}\[site]\Site.xml`
Geocortex viewers	`{G_VIEWERS}`	`{GE_SITES}\[site]\Viewers`
HTML5 viewers		`{G_VIEWERS}\[viewer]`
HTML5 viewer virtual directory		`{G_VIEWERS}\[viewer]\VirtualDirectory`
HTML5 viewer configuration files		`{G_VIEWERS}\[viewer]\VirtualDirectory\Resources\Config\Default`
HTML5 viewer offline bundle files		`{G_VIEWERS}\[viewer]\VirtualDirectory\Resources\Bundled`
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`
Silverlight viewers		`{G_VIEWERS}\[viewer]`
Silverlight viewer virtual directory		`{G_VIEWERS}\[viewer]\VirtualDirectory`
Silverlight viewer configuration file		`{G_VIEWERS}\[viewer]\VirtualDirectory\Config\Viewer.xml`
Viewer backups created by the upgrader		Unnamed instance of Essentials: `{GE_INSTALL}\Default\Backups` Named instances of Essentials: `{GE_INSTALL}\[instance]\Backups`

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 **269**.

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 **269**.

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.

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.