ArcGIS Runtime SDK for .NET (iOS)

Table of Contents

Guide . . . 6
Get started
Install the SDK . . . 8
Add ArcGIS Runtime SDK references . . . 13
Develop your first map app . . . 15
Migrate to 100.x from 10.2.x . . . 18
Fundamentals
Loadable pattern for asynchronous resources . . . 26
Maps and scenes . . . 30
Layers and tables . . . 33
Layer types described . . . 39
Features and graphics . . . 56
Symbols, renderers, and styles . . . 62
Symbol types . . . 66
Tasks and jobs . . . 77
Offline . . . 83
Utility networks . . . 87
Geometries . . . 88
Spatial references . . . 95
Local Server . . . 102
Work with maps (2D)
Display a map . . . 104
Build a new map . . . 111
Wraparound maps . . . 118
Save a map . . . 121
Add a layer . . . 124
Work with scenes (3D)
Display a scene . . . 127
Build a new scene . . . 131
Navigate a scene view . . . 135
Add features and graphics to a scene view . . . 139
Follow a graphic in a scene view . . . 144

Copyright © 1995-2020 Esri. All rights reserved.

Display scenes in augmented reality . . . 150
Offline with maps and scenes
Work offline . . . 166
Take a map offline - on-demand . . . 168
Take a map offline - preplanned . . . 178
Take a map offline - ArcGIS Pro . . . 185
Open an offline map . . . 186
Update an offline map . . . 189
Finish using an offline map . . . 192
Work with offline layers . . . 194
Take a layer offline . . . 198
Take a scene offline - ArcGIS Pro . . . 208
Display information
Add graphics overlays to your app . . . 211
Add graphics and text to graphics overlays . . . 213
Add image overlays . . . 219
Symbolize data . . . 223
Add labels . . . 242
Add raster data . . . 248
Display KML content . . . 257
Display a grid . . . 264
Show the device's location . . . 269
Visualize and compare data over time . . . 273
Display electronic navigational charts . . . 276
Display symbols from a style with a dictionary renderer . . . 279
Search
Identify features . . . 286
Search for places (geocoding) . . . 294
Search for related features . . . 303
Edit features
Editing . . . 306
Edit features . . . 309
Edit geometries . . . 318
Relate features . . . 325

Copyright © 1995-2020 Esri. All rights reserved.

Use geodatabase transactions . . . 329
Sync offline edits . . . 332
Edit KML content . . . 336
Route and get directions
Find a route . . . 342
Display driving directions . . . 348
Navigate a route . . . 351
Route to closest location . . . 357
Generate service areas . . . 365
Add StreetMap Premium data . . . 373
Perform analysis
Geoprocessing . . . 379
Run a geoprocessing task . . . 383
Author and publish a geoprocessing model . . . 388
Get feature statistics . . . 397
Make measurements . . . 401
Manage analyses with analysis overlays . . . 409
Analyze visibility in a scene view . . . 411
Trace a utility network . . . 421
Use the cloud and servers
Access the ArcGIS platform . . . 434
Access portal content . . . 439
Search for content . . . 444
Share a portal item . . . 451
Add items to a portal . . . 454
Use the Authentication Manager . . . 458
Design considerations
Performance considerations . . . 469
Release your app
License your app . . . 474
Deploy your app . . . 482
Create a Local Server deployment . . . 484
Reference
Release notes . . . 486

Copyright © 1995-2020 Esri. All rights reserved.

System requirements 550
Essential vocabulary 568
Coordinate systems and transformations 586
Local Server geoprocessing tools support 587
Platform error codes 588
Supported languages 603
JSON label class properties 605
Support for the web scene specification 617
Legal 620

Copyright © 1995-2020 Esri. All rights reserved.

Guide

This guide describes how to use version 100.8.0 of ArcGIS Runtime SDK for .NET to create Windows Presentation Foundation (WPF) apps, Universal Windows Platform (UWP) apps, native mobile apps in C# for Android, iOS, and the Universal Windows Platform (UWP) with high-quality interactive mapping, queries, geocoding, routing, data editing, and advanced geographic analysis. A great place to start developing once you've installed the SDK is to develop your first map app or browse the wide variety of samples. For release notes and known issues with this release, see Release notes.

The following videos provide an introduction to some common aspects of working with ArcGIS Runtime.

- Understand the loadable pattern:

  NOT IMPLEMENTED

- Understand spatial references:

  NOT IMPLEMENTED

The several APIs included with the SDK share a common design and implementation that enables the reuse of the application logic between apps built for each platform. Functionally, the SDK combines the powerful, high performance mapping and analysis capabilities of the ArcGIS Runtime with features of the ArcGIS platform that supports connected and disconnected workflows.

ArcGIS Runtime SDK for .NET allows your apps to provide functionality like the following:

- Display an interactive map with rich cartography that contains data from a variety of sources
- Add features, rasters, and basemaps stored locally on a device
- Use a powerful set of tasks that leverage ArcGIS capabilities to analyze your maps and provide information to your users
- Search, query, and identify geographic features using any combination of spatial, attribute, or temporal criteria
- Calculate optimal routes between locations and generate directions
- Enable users to collect and edit data in the field
- Synchronize data collected in the field with your enterprise geodatabase
- Reuse your app's logic across your platform-specific projects

What you get

In addition to the platform-specific APIs (WPF, UWP, Android, iOS, and Xamarin Forms), ArcGIS Runtime SDK for .NET includes the following documentation, samples, and utilities.

- Online developer's guide from where you can download the SDK setup, get information about getting started, and browse conceptual overviews of ArcGIS Runtime SDK for .NET.
- API reference for each platform that describes the details of all public classes and members.
- Open source ArcGIS Runtime SDK Toolkit controls for use in your ArcGIS Runtime projects.
- A comprehensive set of samples for each platform.
- The ArcGIS Runtime SDK for .NET forum where you can ask questions and interact with other ArcGIS Runtime SDK developers.
- An online community of ArcGIS developers, including @ArcGISRuntime on Twitter, ArcGIS developer blogs, and ArcGIS Runtime developer forums.

Copyright © 1995-2020 Esri. All rights reserved.

Get started

Copyright © 1995-2020 Esri. All rights reserved.

Install the SDK

Start here to set up your development environment with ArcGIS Runtime SDK for .NET. This SDK contains APIs for developing apps for Windows Presentation Foundation (WPF), Universal Windows Platform (UWP), Xamarin.Android, Xamarin.iOS, and Xamarin Forms. You can have multiple versions of the SDK installed on the same machine. You can also have multiple ArcGIS Runtime SDKs installed side-by-side. There are two options for working with the SDK in .NET:

- Downloading and installing a Visual Studio Extension (Visual Studio for Windows only)
- Referencing NuGet packages, either from nuget.org or an alternate location. Installing the Visual Studio extension also installs the NuGet packages locally

No license is required to develop ArcGIS Runtime apps; there's no need to authorize your development machine or devices. As a member of the ArcGIS Developer Program, you have a free ArcGIS Developer Subscription (Essentials Plan). With this plan you can download and install any ArcGIS Runtime SDK and immediately have access to all the functionality of the APIs for development purposes. You can become a member of the developer program for free.

Download the Visual Studio Extension

ArcGIS Runtime SDK for .NET provides a Visual Studio Extension (.vsix file) with project templates and NuGet packages for supported platforms. Installing the extension will add the Visual Studio templates and configure a local NuGet package source. If you want to add the ArcGIS Runtime NuGet packages from an online source (NuGet.org), there is no need to download and install them locally. The Visual Studio templates, however, are only available by installing the extension (the extension is not available on Visual Studio for Mac).

1. Make sure your development computer meets the system requirements.
2. Download the Visual Studio Extension from the Downloads page. You will need to log in with an ArcGIS developer account. Sign up for a free account if you don't already have one.
3. Save the file to a location accessible from your development machine.

Install ArcGIS Runtime SDK for .NET Visual Studio Extension

Note:
The extension installs NuGet packages and Visual Studio templates for all supported platforms. Depending on the configuration of your development machine, you may not be able to develop for all of these platforms (a Mac computer is required for developing iOS apps, for example). For details, see the System requirements topic.

1. Double-click the file (downloaded previously) to extract the setup files from the archive and start the install.
2. Chose the products you want to install the extension to (Visual Studio 2019, for example). You will only see supported versions of Visual Studio as options. Click **Install** to accept the license terms.

Copyright © 1995-2020 Esri. All rights reserved.

The install will start for all products you selected.

3. When complete, you will see an installation confirmation for the selected products. Click **Close** and make sure to close and restart any open instances of Visual Studio.
4. To use a project template from the extension, start Visual Studio and choose to create a new project. For the supported project types, you will see an ArcGIS Runtime SDK template.

5. To add a local NuGet package, open the **NuGet Package Manager** (**Project > Manage NuGet Packages**, for example). Choose the **Browse** tab in the manager and the local **Esri** package source (configured by the Visual Studio Extension installer). Select the appropriate package for your project and click **Install** to add the package to your project.

Copyright © 1995-2020 Esri. All rights reserved.

Referencing online ArcGIS Runtime SDK NuGet packages

ArcGIS Runtime SDK NuGet packages are hosted on NuGet.org. If you have an internet connection, you can access these packages and add them to your Visual Studio project without having to install anything locally.

Note:

If you need locally available NuGet packages but cannot install the Visual Studio Extension (perhaps you're developing while offline or using Visual Studio for Mac, for example), you can download the packages you need from NuGet.org.

1. Open the **NuGet Package Manager** (**Project > Manage NuGet Packages**, for example).
2. Choose the **Browse** tab in the manager and the **nuget.org** package source. Use the **Search** text box to locate the ArcGIS Runtime packages.
3. Type `ArcGIS Runtime` in the **Search** text box. You will see several matches (for various versions and platforms).

4. Select a version 100.8.0 package for your platform. Click **Install** to add the package to your project.
5. Choose **OK** to proceed with the installation and then **I Accept** to accept the license agreement.

After adding the package, you'll be able to work with the components in ArcGIS Runtime SDK for .NET.

Copyright © 1995-2020 Esri. All rights reserved.

ArcGIS Runtime SDK for .NET Toolkit

The ArcGIS Runtime SDK for .NET Toolkit contains controls and utilities to simplify your app development. For example:

- **Compass**—A control that shows orientation when a map rotated.
- **Legend**—Displays a legend for a single layer in your map (and optionally for its sublayers).
- **MeasureToolbar**—Tools for measuring geographic area or distance.
- **PopupViewer**— Allows you to display details and media, edit attributes, geometry and related records, and manage the attachments of features and graphics (popups are defined in the popup property of features and graphics).
- **ScaleLine**—A scale bar that displays the current scale of the map.
- **TableOfContents**(Preview, WPF only)—A tree view that displays all layers contained in a map, optionally including legend information.
- **TimeSlider**— A control that allows the user to interactively define a temporal range (i.e. time extent) and to animate time moving forward or backward. It can be used to manipulate the time extent in a `MapView` or `SceneView`.

The toolkit is an open source project maintained by the ArcGIS Runtime SDK for .NET team. You can include the toolkit in your project as a NuGet package or download the source code from the GitHub repository and build it locally.

What you get

ArcGIS Runtime SDK for .NET contains everything you need to develop ArcGIS Runtime apps.

- APIs: .NET assemblies and controls.
- Visual Studio project templates to give you a head start on your app development.
- Toolkit controls for use in your ArcGIS Runtime projects.
- Online documentation: SDK Guide and API Reference doc for all supported platforms.
- Code samples: Visual Studio sample projects available on GitHub. See the code samples page for more information.

Next steps

You are ready to begin developing ArcGIS applications using ArcGIS Runtime SDK for .NET. Here are a few ways you can get started:

- Work through the Develop your first map app tutorial.
- Explore code samples.
- Explore SDK concepts and features.
- If you want to use StreetMap Premium data (the StreetMap Premium extension), download the demonstration data from the downloads page for development and testing. Please contact Esri Customer Service for access to a region of your choice for development and testing or to licence StreetMap Premium data for deployment.
- If you want to work with Electronic Navigational Charts (ENC), add the Esri.ArcGISRuntime.Hydrography NuGet package to your app.

Related topics

Add ArcGIS Runtime SDK references
Develop your first map app
System requirements

Copyright © 1995-2020 Esri. All rights reserved.

Release notes

Copyright © 1995-2020 Esri. All rights reserved.

Add ArcGIS Runtime SDK references

To work with ArcGIS Runtime SDK for .NET in your Visual Studio (Windows or Mac) project, you need to add the appropriate ArcGIS Runtime API NuGet package. This process is not unique to ArcGIS Runtime SDK for .NET, you may follow a similar workflow for other APIs you want to use in your project.

The following steps guide you through creating a new **Visual Studio** project for your iOS app. You will also learn how to add the proper references for working with ArcGIS Runtime SDK for .NET classes. You can follow these same basic steps for every **ArcGIS Runtime SDK for .NET** project you create.

Tip:

ArcGIS Runtime SDK for .NET Visual Studio Extension provides a project template (for use in Visual Studio on Windows only) for creating your mapping app, called **ArcGIS Runtime App**. See the Install the SDK topic for information about installing the ArcGIS Runtime SDK for .NET Visual Studio Extension.

Create an iOS app

Use Visual Studio to create an iOS app.

1. Open a supported version of Microsoft Visual Studio.
2. Choose **File > New > Project** (or click **New Project** on the **Start** page) to create a project.
3. Click **iOS > iPhone > Blank App (iPhone)** in the **New Project** dialog box template list.

   Tip:

   If you create your app using the **ArcGIS Runtime App** template, you do **not** need to complete the rest of the steps described in this topic. Creating your project from the template adds the appropriate NuGet packages and references as well as a page with a map view containing a single base layer.

4. Choose a folder location for your new project and give it a name.
5. Click **OK** to create the project.
   Your solution opens in Visual Studio and contains a view controller called **ViewController.cs**.

   Next, you'll add the appropriate ArcGIS Runtime SDK for .NET NuGet package for this platform.

Add ArcGIS Runtime SDK NuGet packages

ArcGIS Runtime SDK NuGet packages are hosted on NuGet.org. If you have an internet connection, you can access these packages and add them to your Visual Studio (Windows or Mac) project without having to install anything locally. If you've installed the ArcGIS Runtime SDK for .NET Visual Studio Extension (Windows only), then you also have these packages available in a local NuGet package source (called **Esri**).

1. Open the **NuGet Package Manager** (**Project > Manage NuGet Packages**, for example).
2. Choose the **Browse** tab in the manager. Choose the Esri package source to add locally installed packages. Otherwise, choose the online **nuget.org** package source. NuGet hosts around 100,000 packages, so you'll need to use the **Search** text box to locate the ArcGIS Runtime packages if using the online package source.

Copyright © 1995-2020 Esri. All rights reserved.

3. To locate the ArcGIS Runtime packages, type `ArcGIS Runtime` in the **Search** text box. You will see several matches (for various versions and platforms).

4. Select a version 100.8.0 package for your platform. Click **Install** to add the package to your project.
5. Choose **OK** to proceed with the installation and then **I Accept** to accept the license agreement.

After adding the package, you'll be able to work with the components in ArcGIS Runtime for .NET API in your code.

The process described in this topic is generally the first thing you'll do for each **ArcGIS Runtime SDK for .NET** project you create from a standard Visual Studio template. The details for adding these references differ slightly between SDK platforms.

Related topics

Install the SDK
Develop your first map app
System requirements
Release notes

Copyright © 1995-2020 Esri. All rights reserved.

Develop your first map app

At the heart of most ArcGIS Runtime SDK apps are the `MapView` control and its associated `Map`. These components allow you to easily add a map to your app with a few lines of code. See the Maps topic for more information about maps and map views.

This tutorial guides you along the shortest path possible to get a mapping app up and running with ArcGIS Runtime SDK for .NET. It assumes that you have installed the ArcGIS Runtime SDK Visual Studio Extension as described in the Install the SDK topic.

Prerequisites

This tutorial requires a supported version of Microsoft Visual Studio and the ArcGIS Runtime SDK for .NET Visual Studio Extension. Refer to the appropriate topics in the guide to see if your machine meets the system requirements or if you need help installing the SDK.

No familiarity with Visual Studio or .NET programming is required to complete this tutorial.

Create an iOS app

Use Visual Studio to create an iOS app.

1. Open a supported version of Microsoft Visual Studio.
2. Choose **File > New > Project** (or click **New Project** on the **Start** page) to create a project.
3. Click **iOS** in the **New Project** dialog box template list. Choose the **ArcGIS Runtime App (iOS)** template.
   Creating your project from the template adds the appropriate references and a page with a map view containing a single base layer.
4. Choose a folder location for your new project and name it `MyFirstMapApp`.

5. Click **OK** to create the project.
   The **ArcGIS Runtime App** template references the ArcGIS Runtime API for .NET NuGet package. When you first build the project, the required NuGet package (or packages) will be restored in the project. You can also restore them manually using the NuGet Package Manager.

   Your solution opens in Visual Studio and contains a view controller called **ViewController.cs**. This file defines the UI for the app with a `MapView` control to display a `Map`.

Copyright © 1995-2020 Esri. All rights reserved.

The `MapViewModel` class contains the logic to create a `Map` and expose it as a public property for consumption in the UI (`MapView`). This is a simple example of the **MVVM** pattern that's used to separate business logic in your app from the user interface tier.

6. Click the **Start** button on the **Standard** toolbar (or **Debug > Start Debugging**) to start your app. You'll see a map similar to the following:

By default, the map displays at the full extent of the first (bottom) layer it contains. To show the initial view of the map at a particular location, you can add logic to define a specific extent when the app starts.

In `MapViewModel`, the code that creates the `Map` defines a basemap using `Basemap.CreateStreetsVector`. There are several static helper methods like this on `Basemap` that can be used to create predefined basemaps for use with your `Map`. If you feel like experimenting, try changing the basemap using one of the following static methods on `Basemap`.

- `CreateDarkGrayCanvasVector`
- `CreateImagery`
- `CreateImageryWithLabels`
- `CreateImageryWithLabelsVector`
- `CreateLightGrayCanvas`
- `CreateLightGrayCanvasVector`
- `CreateNationalGeographic`
- `CreateNavigationVector`
- `CreateOceans`
- `CreateStreets`
- `CreateStreetsNightVector`
- `CreateStreetsVector`
- `CreateStreetsWithReliefVector`
- `CreateTerrainWithLabels`

Copyright © 1995-2020 Esri. All rights reserved.

- `CreateTerrainWithLabelsVector`
- `CreateTopographic`
- `CreateTopographicVector`

Note:
The method names ending in `Vector` use vector tile basemaps. Otherwise, the basemaps are based on raster tiles.

That's it. You created a simple map app without writing a single line of code. As a next step, consider exploring the functionality provided by ArcGIS Runtime SDK for .NET by examining the many samples provided with the SDK.

Related topics

Install the SDK
Add ArcGIS Runtime SDK references
System requirements
Release notes

Copyright © 1995-2020 Esri. All rights reserved.

Migrate to 100.x from 10.2.x

When you compile your unchanged 10.2.x application code against 100.x libraries for the first time, you will see compilation errors. Conceptually, you'll need to consider 100.x as a different product from the previous versions of ArcGIS Runtime. In reality, it is the same product, but it will best serve you to think about it as a different API, at least from the beginning. The ways your workflows are achieved has been re-designed in many cases to fit better with the ArcGIS platform and the Web GIS model as it has evolved today.

This topic outlines areas of the API that have undergone considerable changes, and provides guidance for re-factoring 10.2.x code for a 100.x app. Although this topic doesn't cover every change, it will help you get over some major migration hurdles. It will also help you decide whether functionality required for your app is available in 100.x.

Changes to the ArcGIS Runtime SDK for .NET

Since the architectural changes for version 100.x were implemented at the lowest level of ArcGIS Runtime, most of the changes described in this topic apply to all ArcGIS Runtime products. There are some, however, that apply specifically to ArcGIS Runtime SDK for .NET. Many .NET-specific changes are related to the APIs for individual platforms, how they are delivered to your development projects, and so on. For more information related to this release, refer to the Release notes.

- ArcGIS Runtime SDK for Xamarin APIs are now part of the ArcGIS Runtime SDK for .NET. The SDK now contains APIs for the following platforms.
  - Windows Presentation Framework (WPF)
  - Universal Windows Platform (UWP) - Replaces the 10.2.x Windows Store and Windows Phone APIs.
  - Xamarin.Android
  - Xamarin.iOS
  - Xamarin.Forms - Supports cross-platform development for Android, iOS, and UWP.
- Each of the APIs listed above can be added to your project from packages hosted on nuget.org. You do not need to install the SDK to access these packages.
- The SDK can be installed locally on Windows using a VSIX file. This will install the same NuGet packages locally on your machine as well as add Visual Studio project templates. See Install the SDK for more information.
- **Local Server** is now a separate install. When developing an ArcGIS Runtime app, you can use a lightweight client API to work with classes in Local Server. The Local Server components themselves are installed separately with ArcGIS Runtime Local Server SDK. See Installing the Local Server SDK for details.
- The minimum required version of .NET Framework is now **4.5.2**. If you attempt to build your project against an earlier version, you will receive an error like: `Error: the tag 'MapView' does not exist in XML namespace 'http://schemas.esri.com/arcgis/runtime/2013'`
- The process for creating a deployment for your app has been greatly simplified since 10.2.x, and no longer requires an **ArcGIS Runtime .NET Deployment Manifest**. See Deploy your app for details.

Functional changes

In many cases where functionality appears to be missing from 100.x, it is available but has been implemented differently. These changes require you to change your code to use new programming patterns or classes. The following sections describe some of the functional areas of the API where these types of changes have been implemented.

Copyright © 1995-2020 Esri. All rights reserved.

Maps and scenes

In ArcGIS Runtime SDK for .NET, at 10.2.7, the `Map` and `Scene` objects were split out from the controls that display them (`MapView` and `SceneView`) in order to facilitate a Model-View-ViewModel (MVVM) architecture for your apps. At 100.0.0, this change was adopted by all ArcGIS Runtime SDKs. If your app was created with ArcGIS Runtime SDK for .NET 10.2.7, this will not affect your migration to 100.x.

Views

The `GeoView` base class, inherited by `MapView` and `SceneView`, is solely responsible for display and interaction, separating concerns from the model objects (`Map` and `Scene`) and allowing the APIs to be simplified and harmonized between the 2D and 3D worlds. `GeoView` contains graphics overlays, as well as operations, to easily identify features and graphics without having to write any layer type specific code. At 100.0.0, this change was adopted by all ArcGIS Runtime SDKs. If your app was created with ArcGIS Runtime SDK for .NET 10.2.7, this will not affect your migration to 100.x.

Loadable

The `ILoadable` was introduced at 100.0.0. Its design and purpose is intended for those workflows that involve accessing lots of data over the wire from connected online resources. All resources that load metadata asynchronously to initialize their state (such as maps, layers, and tasks) adopt the loadable pattern. The loadable pattern makes the behavior of loading state more consistent and explicit. Loadable resources do not automatically load their state. They load lazily when loaded by your app code or loaded by other objects that depend on them. This becomes useful for cases where, for example, you need to obtain information from a map resource without having to visualize it first. You can explicitly load the map resource without first associating it to a map view. The status of a loadable resource is easy to monitor to determine whether it is loading, has loaded successfully, or has failed to load. Your app can retry loading it if the resource failed to load due to a transient or recoverable condition.

Common use cases for using `Loadable` are explained and demonstrated through code examples in the topic Loadable pattern for asynchronous resources.

If 10.2.x versions of your apps performed editing tasks involving feature services, it's important for you to fully understand the new loadable pattern. For example, retrieving features from an `ArcGISFeatureTable` return `ArcGISFeature` object which implements the loadable pattern for increased efficiency. When getting `ArcGISFeature` objects for rendering and query purposes, a minimum set of required fields is returned, such as identifiers, geometry, fields used for symbolizing, and so on. You must first load the feature that you want to edit, or the edit operation will fail. For complete details and code examples of loading features in editing workflows, see the Edit features topic.

Graphics overlays and graphics

Graphics are used to display temporary or ad-hoc geographic data on top of a map. In 10.2.x, you could add graphics to your map using a `GraphicsLayer`. At 10.2.7, the introduction of `GraphicsOverlay` allowed graphics to be displayed as part of the map view or scene view. `GraphicsOverlay` ensures that graphics are always displayed on top of everything else in the view, even when layers are reordered or the map is changed. Graphics in an overlay are also reprojected, conforming to the spatial reference assigned to the view. Starting at 100.0.0, graphics may **only** be displayed in a `GraphicsOverlay`, `GraphicsLayer` is no longer available.

The guide topic Add graphics overlays to your app provides code for creating graphics overlays and adding them to the map. You can see the code for adding and working with graphics in a graphics overlay in Add graphics and text to graphics overlays.

Copyright © 1995-2020 Esri. All rights reserved.

Also at 100.x, there is a easier pattern for identifying graphics within graphics overlays. The `IdentifyGraphicsOverlayAsync` method on `GeoView` (`MapView` or `SceneView`) identifies visible graphics in the specified graphics overlay, near the provided screen point. The Identify graphics code sample illustrates this technique.

Feature tables

`ServiceFeatureTable` now has a `FeatureRequestMode` that is like the 10.2.x `Mode` property (`Snapshot` or `OnDemand`). The value determines how features are requested from the service for use in the client.

- **On Interaction with Caching** - Features are requested and stored in the table when needed based, on user interaction (panning or zooming to different extents, for example). Features are cached to reduce the number of requests for features.
- **On Interaction without Caching** - Features are always requested from the service. There is no caching of features, so this consumes the largest amount of network bandwidth.
- **Manual Caching** - Features are only requested when explicitly requested by the app. All queries are made to the local version of the data.

`ServiceFeatureTable` no longer has `Where` or `OutFields` properties. To filter the data returned from the service, you can call `PopulateFromServiceAsync` with values for those parameters.

Identify features and graphics

`FeatureLayer` and `GraphicsOverlay` no longer have a `HitTestAsync` method for identifying features or graphics (`GeoElement` objects) that intersect a geometry. Instead, you should use one of the following methods on `GeoView` to do the equivalent.

- `IdentifyGraphicsOverlayAsync` - Identify graphics from a single `GraphicsOverlay` in the view.
- `IdentifyGraphicsOverlaysAsync` - Identify graphics from all `GraphicsOverlay` objects in the view.
- `IdentifyLayerAsync` - Identify features from a single `Layer` in the view.
- `IdentifyLayersAsync` - Identify features from all `Layer` objects in the view.

See Identify graphics for an example of using these methods.

Geoprocessing

The 10.2.x `Geoprocessor` class has been replaced with `GeoprocessingTask` in 100.x. The task can create a geoprocessing job that tracks progress of the server processing. You can listen for status changes as the job progresses, and respond when it's complete by reading the results (outputs) from the job. The process is outlined in the following steps.

1. Create a new `GeoprocessingTask`. Pass in the Uri for the geoprocessing service endpoint.
2. Create a new `GeoprocessingParameters` object to store the inputs to the task.
3. Call `GeoprocessingTask.CreateJob` to return a `GeoproccessingJob`. Pass in the `GeoprocessingParameters`.
4. Set up a listener for the `GeoproccessingJob.JobChanged` event.
5. Start the job by calling `GeoproccessingJob.Start`.
6. In the `JobChanged` handler, check the job status.

Copyright © 1995-2020 Esri. All rights reserved.

7. When the status is `JobStatus.Succeeded`, call `GetResultAsync` to get the results, which includes a dictionary of outputs and (optionally) a map image.

See the Geoprocessing topic for more information and examples.

Hydrography

At 100.2.0, a new `EncLayer` displays content from ENC (Electronic Navigational Charts) data in S-57 format. In addition to displaying and identifying features, you can change various display settings for view groups, text, and other elements (such as isolated dangers, contours, color scheme, and so on). In 10.2.x, this functionality is available in the `Esri.ArcGISRuntime.Hydrographic` namespace. At 100.2.0, you'll find the same API in `Esri.ArcGISRuntime.Hydrography`.

Offline

One of the big differences with 100.x is that the APIs for common operations such as editing, searching, geocoding, or routing are the same whether you are online or offline. You just need to specify whether your data source.

With respect to packaging maps and taking them offline, ArcGIS Pro lets you create and share offline mobile map packages. See Share a mobile map package. A Mobile Map Package (.mmpk), represented by the `MobileMapPackage` class, is a set of items bundled together into a single archive file for easy transport. The items are one or more maps, their associated layers and data, and optionally networks and locators. A mobile map package also includes metadata about the package that you can glean useful information from using the new API. You don't need a Local Server to access these packages. You can also take the map offline directly using the on-demand and preplanned workflows provided since 100.3.

See the `MobileMapPackage` class in the API reference and the Work with offline maps topic for a thorough description of how to create and work with Mobile Map Packages.

You can also take your connected ArcGIS based feature and tiled layers offline on demand with dedicated **tasks** and associated **jobs**, just as you were able to with previous versions. However, you might find that the class names and methods differ slightly from previous versions. The `GeodatabaseSyncTask` works with ArcGIS feature services to take features offline in a mobile geodatabase (.geodatabase file) and allow them to be edited and synced. The `ExportTileCacheTask` extracts tiles from a tiled ArcGIS map service as a tile package (.tpk file) and allows them to be viewed offline.

The pattern for generating a local geodatabase has changed at 100.x and you'll need to make some changes to this code if migrating a 10.2.x app.

1. Call `GeodatabaseSyncTask.CreateAsync` to create an instance of the `GeodatabaseSyncTask`. Pass in the Uri for the feature service endpoint.
2. Set some of the same 10.2.x options, such as `ReturnAttachments`, `OutSpatialReference`, and `SyncModel`.
3. Call `GeodatabaseSyncTask.GenerateGeodatabase` to return a `GenerateGeodatabaseJob`.
4. Set up a listener for the `GenerateGeodatabaseJob.JobChanged` event.
5. Start the job by calling `GenerateGeodatabaseJob.Start`.
6. In the `GenerateGeodatabaseJob.JobChanged` handler, check the job status.
7. When the status is `JobStatus.Succeeded`, read the contents of the database, add feature layers to the map, and so on.

The Edit features guide topic provides code examples and in-depth discussion for editing both online and offline with the new API.

Copyright © 1995-2020 Esri. All rights reserved.

Authentication

At 100.x, the `IdentityManager` class has been renamed `AuthenticationManager`, but has the same essential functionality.It allows the developer to decide how much control to exercise over authentication-related events handled by an app. It also caches credentials by default, reducing the number of times a user is prompted for credentials. The authentication manager issues an authentication challenge whenever authentication-related events occur. Developers can monitor these challenges and respond with appropriate credentials to get access to secured resources, or allow the authentication manager to prompt the end user for credentials.

One change you will notice from the previous `IdentityManager` is that `Uri` objects are generally now used to represent URL endpoints (instead of using URL strings). `ServerInfo.ServerUri` is now set with a `System.Uri` rather than a string. Likewise, `AuthenticationManager.FindCredential`, `AuthenticationManager.GenerateCredential`, and `IOAuthAuthorizeHandler.AuthorizeAsync` take `Uri` objects as arguments instead of strings.

For some examples of using `AuthenticationManager` to handle authentication in your runtime app, see the Author a map and Token authentication challenge samples.

Portal

The mapping API has been refactored for better integration with the portal API. This API allows you to load maps stored in a portal (web maps) into your app, make updates to web maps, and save changes back to the portal item. Also, your app can save maps created by your app as a new portal item. These web maps can then be used elsewhere within the ArcGIS system.

The `WebMap` and `WebMapViewModel` classes are no longer used to load a web map from a portal. At 100.x, the `Map` class has constructors that take either a `PortalItem` or a `Uri` representing the web map.

The Save a map topic provides great examples for using the new map saving capabilities. For an example of saving and updating web maps stored in your portal, see the Author a map sample.

Note:
Many of the classes in the `Esri.ArcGISRuntime.Portal` namespace have changed at 100.x. See the **Minor changes** section for details.

Error handling

With 100.x, it's easier to determine where errors occur in the stack so you can provide better error messages to your Runtime app users. A new standardized error domain property indicates whether the error occurred client-side within the ArcGIS Runtime, or server-side from an ArcGIS Server service or web service. A consistent error code property can be used to further diagnose the error's cause or determine what error message should be displayed to the user. For a complete list of error domains and codes, see the Platform error codes topic.

Licensing changes

Like earlier, you have access to all the capabilities during development and testing, however the licensing model for deployment has changed to include 4 levels - Lite, Basic, Standard, and Advanced where each level unlocks a different set capabilities. You can license your app either by using a license key or a by logging into a portal with a named user. More details are available in the License your app topic.

Copyright © 1995-2020 Esri. All rights reserved.

Minor changes

Many of the breaking changes you will encounter when migrating your app from 10.2.x to 100.x are due to classes moving to a different namespace or changes to class or member names. This section describes some of the reorganizing and renaming in the API between these versions.

Namespace changes

Some of the namespaces that were available in 10.2.x of ArcGIS Runtime SDK for .NET have been removed. Usually, the classes that existed in these namespaces have been moved elsewhere. Sometimes, the same functionality exists at 100.x but has been implemented in a new class.

- `Esri.ArcGISRuntime.Layers` no longer exists. The classes found here at 10.2.x are now in `Esri.ArcGISRuntime.Mapping`.
- `Esri.ArcGISRuntime.Controls` no longer exists. Classes like `MapView` and `SceneView` are now in `Esri.ArcGISRuntime.UI.Controls`. Others, such as `GraphicsOverlay` and map grids are now in `Esri.ArcGISRuntime.UI`. Things like `Map` and `Camera` have been moved to the new `Esri.ArcGISRuntime.Mapping` namespace.
- `Esri.ArcGISRuntime.Symbology.SceneSymbology` no longer exists. The corresponding 100.x classes are in `Esri.ArcGISRuntime.Symbology`.
- `Esri.ArcGISRuntime.WebMap` no longer exists. Many of the classes found here at 10.2.x have been moved to `Esri.ArcGISRuntime.Mapping` and `Esri.ArcGISRuntime.Mapping.Popups`.
- At 10.2.x, the classes for working with Electronic Navigational Charts (ENC) are in the `Esri.ArcGISRuntime.Hydrographic` namespace. In 100.2.0, you'll find them in `Esri.ArcGISRuntime.Hydrographic` (as well as an `EncLayer` in `Esri.Mapping`).

Class and member name changes

Several classes, properties, and methods have been renamed at 100.x for clarity and succinctness. The following table lists some naming changes in the API.

10.2.x	100.x
Layers	
`ArcGISTiledMapServiceLayer`	`ArcGISTiledLayer`
`ArcGISLocalTiledLayer`	`ArcGISTiledLayer`
`SceneLayer`	`ArcGISSceneLayer`
`IdentityManager`	`AuthenticationManager`
`Layer.ID`	`Layer.Id`
`Layer.DisplayName`	`Layer.Name`
`ArcGISTiledMapServiceLayer.ServiceUri` (string)	`ArcGISTiledLayer.Source` (`Uri`)
`ArcGISLocalTiledLayer.Path` (string)	`ArcGISTiledLayer.Source` (`Uri`)
MapView	

Copyright © 1995-2020 Esri. All rights reserved.

10.2.x	100.x
Events: `MapViewTapped`, `MapViewDoubleTapped`, and so on.	These events are now defined on the `GeoView` base class: `GeoViewTapped`, `GeoViewDoubleTapped`.
`MapView.ExtentChanged`	`GeoView.ViewPointChanged`
`Extent` (`Envelope`)	`VisibleArea` (`Polygon`)
`Editor`	`SketchEditor`
`DrawShape` enum (defines shape type for `Editor` to draw).	`SketchCreationMode`
`MinScale`, `MaxScale`	These properties are now defined on the `Map`: `Map.MinScale`, `Map.MaxScale`
`WrapAround` (`bool`)	`WrapAroundMode` (`WrapAroundMode` enum)
`LayerLoaded`	`LayerViewStateChanged`
Portal	
`ArcGISPortalItem`	`PortalItem`
`ArcGISPortalUser`	`PortalUser`
`ArcGISPortalGroup`	`PortalGroup`
`QueryParameters` `SearchParameters`	`PortalQueryParameters`
`SearchResultInfo<T>`	`PortalQueryResultSet<T>`
`ArcGISPortal.CurrentUser`	`ArcGISPortal.User`
`ArcGISPortal.SearchItemsAsync`	`ArcGISPortal.FindItemsAsync`
Symbology and renderers	
3D Marker symbols, such as `BoxMarkerSymbol`, `SphereMarkerSymbol`, and so on.	Use `SimpleMarkerSceneSymbol` and set the `Style` using a value from the `SimpleMarkerSceneSymbolStyle` enum.
`ClassBreaksRenderer.Field`	`ClassBreaksRenderer.FieldName`
`ClassBreaksRenderer.Infos` (`ClassBreakInfo` collection)	`ClassBreaksRenderer.ClassBreaks` (`ClassBreak` collection)
SceneView and Scene	
`Scene.Surface`	`Scene.BaseSurface`
`ServiceElevationSource`	`ArcGISTiledElevationSource`
`FileElevationSource`	`RasterDataSource`
`SceneView.ElevationExaggeration`	`Surface.ElevationExaggeration`

Copyright © 1995-2020 Esri. All rights reserved.

Fundamentals

Copyright © 1995-2020 Esri. All rights reserved.

Loadable pattern for asynchronous resources

Resources such as layers, maps, portal items, tasks, and so on, commonly rely on remote services or datasets on disk to initialize their state. The nature of accessing such data requires the resources to initialize their state asynchronously. The loadable design pattern unifies the behavior that different resources use to load metadata asynchronously, and resources that adopt this pattern are referred to as "loadable." The pattern also provides a mechanism to retry if previous attempts to load failed so that you can properly handle and recover from exceptional circumstances such as network outages or service interruption. Loadable resources appropriately handle concurrent and repeated requests to load in order to accommodate the common practice of sharing the same resource instance among various parts of an application, and also permit cancellation so that you can cancel loading a resource, for example, if the service is slow to respond. Finally, loadable resources provide information about their initialization status through explicit states that can be inspected and monitored.

NOT IMPLEMENTED

Loadable API

Classes that conform to the loadable pattern implement the `ILoadable` interface.

Load status

The `LoadStatus` property reflects the state of the loadable resource. Four states are possible.

- `NotLoaded`—the resource has not been asked to load its metadata and its state isn't properly initialized yet.
- `Loading`—the resource is in the process of loading its metadata asynchronously.
- `FailedToLoad`—the resource failed to load its metadata (for example, due to network outage, or the operation was cancelled, and so on.) The error encountered is available in the `LoadError` property.
- `Loaded`—the resource successfully loaded its metadata and its state is properly initialized.

The following state transitions represent the stages that a loadable resource goes through.

Copyright © 1995-2020 Esri. All rights reserved.

The `ILoadable` interface includes events for responding to changes in a resource's load status.

Loading

A resource commences loading its metadata asynchronously when `LoadAsync` is invoked. At that time, the load status changes from `NotLoaded` to `Loading`. When the asynchronous operation completes, the `Loaded` callback is invoked. If the operation encounters an error, the object's `LoadError` property is populated, and load status is set to `FailedToLoad`. If the operation completes successfully, `LoadError` is null and the load status is set to `Loaded`, which means the resource has finished loading its metadata and is now properly initialized.

Many times, the same resource instance is shared by different parts of the application. For example, a legend component and a table of contents component may have access to the same layer, and they both may want to access the layer's properties to populate their UI. Or the same portal instance may be shared across the application to display the user's items and groups in different parts of the application. In order to accommodate this type of application development pattern and make things simple for you, `LoadAsync` supports multiple "listeners". It can be called concurrently and repeatedly, but only one attempt is ever made to load the metadata. If a load operation is already in progress (`Loading` state) when `LoadAsync` is called, it simply piggy-backs on the outstanding operation and the callback is enqueued to be invoked when that operation completes. If the operation has already completed (`Loaded` or `FailedToLoad` state) when `LoadAsync` is called, the callback is immediately invoked with the past result of the operation, be it success or failure, and the state remains unchanged. This makes it safe for you to liberally invoke `LoadAsync` on a loadable resource, without having to check if the resource is already loaded or not, and without worrying that it will make unnecessary network requests every time.

If a resource has failed to load, calling `LoadAsync` on it subsequently will not change its state. The callback will be invoked immediately with the past load error. In order to retry loading the resource, you must use `RetryLoadAsync` instead. The advantage of this approach is that if, hypothetically, `LoadAsync` were to retry loading a failed resource, an app could easily end up making many asynchronous requests to load the same resource just because it shared the same resource instance in

Copyright © 1995-2020 Esri. All rights reserved.

different parts of the application and each part tried to load the resource without first checking if it was already loading. This pattern allows `LoadAsync` to be invoked indiscriminately without worrying that it might add overhead, and makes retrying a separate and explicit scenario using `RetryLoadAsync`.

Retry loading

A resource retries to load its metadata when `RetryLoadAsync` is invoked, but only if it previously failed to load (`FailedToLoad` state) or wasn't loaded to begin with (`NotLoaded` state). The resource transitions to the `Loading` state and makes a new attempt to fetch its metadata. The `Loaded` event callback is invoked when the asynchronous operation completes.

If the resource has already fetched its metadata and has initialized its state (`Loaded` state), retry doesn't do anything. Instead, retry immediately calls the callback with the past result and the state of the object remains unchanged. The object's metadata isn't fetched again.

If an asynchronous operation is already in progress (`Loading` state) when `RetryLoadAsync` is called, it simply piggy-backs on the outstanding operation and the callback is enqueued to be invoked when that operation completes.

The main use case for this method is if the loadable failed to load previously, for example, due to network outage or service interruption. It is not meant to refresh the metadata for an already loaded resource which should instead be accomplished by creating a new instance of the loadable resource.

Cancel loading

A resource cancels any outstanding asynchronous operation to load its metadata when `CancelLoad` is invoked and it transitions from `Loading` to `FailedToLoad` state. The `LoadError` property will contain information that reflects that operation was cancelled.

This method should be used carefully because all enqueued callbacks for that resource instance will get invoked with an error stating that the operation was cancelled. Thus, one component in the application can cancel the load initiated by other components when sharing the same resource instance.

The `CancelLoad` method does nothing if the resource is not in `Loading` state.

Conveniences

Cascading load dependencies

It is common for a loadable resource to depend on loading other loadable resources to properly initialize its state. For example, a portal item cannot finish loading until its parent portal finishes loading. A feature layer cannot be loaded until its feature service table is first loaded. This situation is referred to as a load dependency.

Loadable operations invoked on any resource transparently cascade through its dependency graph. This helps simplify using loadable resources and puts the responsibility on the resource to correctly establish and manage its load dependencies.

The following code example shows how this cascading behavior leads to concise code. Loading the map causes the portal item to begin loading, which in turn initiates loading its portal. You do not have to load each one of them explicitly.

```
// creating a portal instance (pass in token credentials)
var myPortal = await ArcGISPortal.CreateAsync(new Uri(portalUrl), CancellationToken.None, token);

// get a portal item (using its ID)
var webMapItem = await ArcGISPortalItem.CreateAsync(myPortal,
"5cdcac1d6d814b82a48453d3e3876fd1");
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// create a new map from the item, load the map
var myMap = new Map(webMapItem);
await myMap.LoadAsync();

// make sure the map loaded
if (myMap.LoadStatus != LoadStatus.FailedToLoad)
{
    // add the map to the map view (dispatch to the UI thread)
    var t = new System.Threading.Thread(new System.Threading.ThreadStart(() => {
        InvokeOnMainThread(() => {
            MyMapView.Map = myMap;
        });
    }));
    t.Start();


}
```

It is possible that dependencies may fail to load. Some dependencies might be critical without which the initiating resource cannot load successfully, for example, a portal item's dependency on its portal. If a failure is encountered while loading such a dependency, that error would bubble up to the resource that initiated the load cycle, which would also fail to load. Furthermore, retrying to load that resource would automatically retry loading the failed dependency. Other load dependencies may be incidental, such as a map's dependency on one of its operational layers, and the resource may be able to load successfully even if one of its dependency fails to load.

Overriding state before initialization

A loadable resource's state is not properly initialized until it has finished loading. Accessing properties of the resource before the resource is loaded could return null or uninitialized values that might change when the resource finishes loading. Therefore, it is always advisable to wait until the resource has finished loading before accessing its properties. However, many times, especially while prototyping, you may want to change a property value before the resource is loaded without regard for its proper initialized value. For instance, you may want to change the scale visibility of a layer or the initial viewpoint of a map. To simplify this workflow without requiring that the resource first be loaded, loadable resources permit overriding their properties before the resource has finished loading and the overridden value will take precedence over the value specified in the resource's metadata.

Copyright © 1995-2020 Esri. All rights reserved.

Maps and scenes

Maps (2D) and Scenes (3D) are interactive displays of geographic information that you can use to tell stories and answer questions. The resources and configuration that define a map or scene are gathered together into a single document (container) which can be opened, shared, and edited across the ArcGIS platform. This topic describes how maps and scenes fit into the ArcGIS platform and your own apps.

Maps and scenes

Maps and scenes can consist of the following components:

- **Basemap & elevation sources**- Both maps and scenes use basemaps to provide context (the background) for the geographic content in your app. Elevation sources allow scenes to render realistic terrain.
- **Operational layers** - These visualize the important content in your map or scene - that which informs the user, tells a story, or otherwise provides value. To learn more about the role of layers and tables, see Layers and tables.
- **Data tables** - Data that is visualized in a layer can be related to non-geographic data. For example, a layer showing employee locations might be related to a non-spatial table that contains employees and their roles. This table can be stored alongside the explicitly geographic content of the map. To learn more about the role of layers and tables, see Layers and tables.
- **Locators & network datasets** - Maps can contain information about transportation networks. This allows you to build geocoding, routing, service area analysis, and other network analysis capabilities into your app. For example, the StreetMap Premium offline mapping and routing product uses a mobile map package (a map packaged for offline use) to deliver both offline maps and offline routing.
- **Bookmarks** - Viewpoints that are stored alongside the map. For example, you might have a bookmark for each mountain in a map detailing a climber's adventures.
- **Transient graphics** - Maps and scenes can display temporary graphics from a GPS, web socket or user input using a graphics overlay. To discover more, see Features and graphics.

If you want to display a map or scene in your app you can use a `MapView` to display maps and a `SceneView` to display scenes. In this context, the map or scene represents the model, while the `MapView` or `SceneView` is the view tier in the model-view-controller pattern. These views handle rendering the map or scene and providing services to navigate the scene, show overlay graphics, enable identify and selection, and perform visual analysis.

Map and map view objects allow you to build new maps, display maps, save maps, and share maps. For more information, see:

- Build a new map
- Display a map
- Save a map
- Share a portal item

Scene and scene view objects allow you to build new scenes and display scenes. See Build a new scene and Display a scene.

Also see the API reference for more information.

Maps and scenes compared

A 2D map resembles a paper map where a representation of the earth is projected onto a 2D surface. In 3D, the map is projected onto a globe that more accurately models the spherical shape of the earth.

Copyright © 1995-2020 Esri. All rights reserved.

In 2D, any point on the globe is represented by an x and y coordinate (longitude and latitude). In 3D, points on the globe are represented with three coordinates: x (longitude), y (latitude), and z (elevation). See Z-values and 3D for more information about this topic.

3D scenes require configuring a camera (see Camera Class to set a viewpoint.

A camera has:

- a location (x longitude, y latitude)
- altitude (height, in meters, from sea level)
- heading (angle about the z axis the camera is rotated, in degrees)
- pitch (angle the camera is rotated relative to the y axis, in degrees)
- roll (angle the camera is rotated about the axis that's perpendicular to the camera's direction, in degrees)

The camera also includes a CameraController (see CameraController Class), which provides specialized camera navigation.

3D scenes may contain 3D objects, such as buildings, trees, and other objects. 3D objects can be represented with 3D symbols or SceneLayers. Textures are draped over 3D objects.

3D scenes may require different UI controls to navigate the scene. Navigating with a camera offers different parameters than does moving a viewpoint, such as altering the camera altitude, heading, and pitch.

Due to the nature of 3D rendering, there are additional properties for scenes that do not apply to 2D maps, such as surface placement with LayerSceneProperties and extrusion with RendererSceneProperties.

Note:

- When you view or analyze spatial data from different layers, you must always consider their spatial reference. For details, see Spatial references.
- Objects that can be symbolized on your map can come from persisted data (feature data, such as roads and parcels in a geodatabase) and transient data (graphics, such as data coming through a web socket). Military symbology can be symbolized as either of these. For details, see Features and graphics, Symbols and renderers, and Display military symbols with a dictionary renderer.

Maps and scenes can both operate in situations where your device is fully online, occasionally offline or fully offline.

Online with web maps and web scenes

Online maps and scenes, or web maps and web scenes, allow geographic information to be created, edited and shared across the ArcGIS platform. It is easy to share online maps and scenes because you simply provide a hyperlink to the web map or web scene and launch it from within your ArcGIS Runtime app.

To begin creating your own online maps, see the ArcGIS Online Help topic Get started with maps. ArcGIS Online also provides a Living Atlas of the World with authoritative maps on many topics. Here, you can explore maps and data from Esri and thousands of other organizations, and combine them with your own data to create new maps.

To learn how to create online scenes, see the ArcGIS Online Help topic Get started with scenes.

Offline with mobile map and scene packages

Users can take maps and scenes offline using mobile map packages and mobile scene packages. These packages let you continue working with maps or scenes while your app is disconnected.

Copyright © 1995-2020 Esri. All rights reserved.

A mobile map package contains the definition of one or more maps along with any simple features, raster datasets, tables, relationship classes, tile packages, or vector tile packages the maps require. Mobile scene packages also contain the definition of one or more scenes along with the data they require. A mobile scene package can also contain elevation data if the scene needs to drape its data across a 3D terrain. You can create read-only mobile map and scene packages using ArcGIS Pro, and you can create editable mobile map packages using the Services pattern available with the ArcGIS Runtime.

ArcGIS Pro

Using ArcGIS Pro lets you create read-only mobile maps or scenes that can be distributed to a large number of users. Typically these users and their devices operate in a fully disconnected environment.

ArcGIS Pro provides two geoprocessing tools: **Create Mobile Map Package** and **Create Mobile Scene Package**. For more details, see Create a Mobile Map Package and Create a Mobile Scene Package in Package toolset in the ArcGIS Pro help . The maps or scenes and their data are stored in a single archive file called a mobile map package (*.mmpk) or mobile scene package (*.mspk) file. The advantage of this approach is that you can create one mobile package file and distribute it to many devices. You can distribute these files by copying them onto a device, or uploading them to your organization's portal for users to download.

Services Pattern

This Services pattern is appropriate for occasionally connected devices where the app allows the user to edit map data and share their changes with other users when connectivity is restored.

The services patterns allows you to build ArcGIS Runtime apps to download a mobile map package, in a directory format, to a device. This is possible as long as the map and its layers have been authored for offline use (see Take web maps offline in Use and edit data in the ArcGIS Pro help). Once disconnected, the user can edit the feature data and then synchronize any changes when network connectivity is restored. Runtime offers two main workflows:

- **Preplanned offline workflow** - A field worker can download an offline map that has been generated by the map author, making use of a pre-planned offline workflow. See Preplanned offline workflow.
- **On-demand offline workflow** - The field worker can select an area of interest on a map, generate an offline map for that area and download it to their device. See On-demand offline workflow for more information.

In both cases, the author of the web map can specify whether the mobile map uses the basemap defined by the web map or a tile package that is already on the device. Using a local tile package already on the device will reduce the size of the mobile map package, and can decrease its download time. These tile packages can be located using an actual file path on the device or a path that is relative to the mobile map package.

Related topics

Display a map
Build a new map
Wraparound maps
Save a map

Copyright © 1995-2020 Esri. All rights reserved.

Layers and tables

A layer displays geographic data in a map or scene. See Maps and scenes are containers for layers and tables. A table is a source of data; geographic data in tables can be consumed by some types of layers. ArcGIS Runtime supports many layer types. See Layer types described.

Layers vary in their implementation styles and capabilities. For example, feature layers are conceptually tied to an underlying table; data from the table is symbolized by a renderer. ArcGIS map image layers and web map service (WMS layers) show map images rendered by a server on demand. ArcGIS tiled layers and web tiled layers show tiles that were rendered ahead of time by the server. ENC layers show data from files in device storage.

Layers can have one of two roles in your map or scene:

- **Basemap layers** provide context for the information in your map or scene. For example, the imagery or street map backgrounds on consumer mapping services are basemaps.
- **Operational layers** contain the primary content of the map or scene. For example, a layer with all of the gas stations nearby would be an operational layer.

Runtime layers can be used as operational or basemap layers; you have the flexibility to use layers in the way that makes sense for your app and cartography.

Layers define how information is displayed. They can consume data from tables, local files, or remote services. If you only need to put points on a map, or work with temporary data without an underlying model, consider using graphics instead. See Features and graphics.

Many layers can be shown in a map or scene. You can control the draw-order of layers as well as their visibility and opacity. Operational layers may be added to a group layer to control their visibility as a group. See Group layer.

There are many layer types available. See Layer types described to learn about ArcGIS Runtime layers and how to choose the right layer types for your map or scene.

Common layer properties

All layers support the following common properties:

- **Name** — a human-readable name
- **Description** — a human-readable description of the layer's content
- **Extent** — the geographic area that contains the layer's content
- **Spatial reference** — defines how coordinates should be interpreted. See Spatial references for more information.
- **Visibility** — lets you hide or show the layer without removing it from the map or scene.
- **Scale range** — controls layer visibility based on how far the user has zoomed into the map or scene using the `MinScale` and `MaxScale` properties.
- **Opacity**

Note that layers vary significantly in their implementation styles and capabilities. Most layers have many more properties than are shown in the list above. See Layer types described for details.

Choosing layers for your map or scene

When it comes time to choose layer types for your app, consider the requirements you established for how your app will work. Ask yourself the following questions:

Copyright © 1995-2020 Esri. All rights reserved.

- What information is your app designed to display or analyze? Does your app process or visualize two-dimensional (2D) or three-dimensional (3D) data or both? Some layer types are designed specifically to be sourced from with 3D data sources like elevation surfaces and point clouds, others are sourced with 2D data like rasters. Nevertheless, in many cases 2D or 3D data may be used in either a map or a scene.
- Do users need to compare data with a time component? Some layers are time-enabled, meaning that the content can be filtered using a time extent that depends on a date/time attribute in the layer.
- Do users have an always-on network connection? Without an always-on connection, some data is usually cached by the app or in the local file system.
- Do users need to be able to query, analyze or select data? This determines what attributes each features in the layer should have.
- Do users need to be able to edit layer content? This has an impact on the source of the layer's data, and the editing workflow.
- Do you or your users need control over the style (colors, fill patterns, icons, etc.) of the layer? Some layers allow changes to styling, others use only the styling provided by the layer's source.
- Does your app need to interoperate with other systems or ArcGIS Portal? This impacts your choice when layer data could come from different source types (such as ArcGIS Portal, services or local data) or will be shared out from your app to other systems or ArcGIS Portal.

After you understand your requirements, consult the table in Layer types described to see which layers meet your needs.

Understanding your data is the first step in the 'Data, Design, Develop' pattern for creating ArcGIS apps. ArcGIS DevLabs will walk you through the process of bringing data for your apps into ArcGIS.

If you want to quickly visualize data that's available to your app, but don't want to store it in an ArcGIS portal or layer, consider using graphics. See Features and graphics.

Choosing basemap layers

The basemap is the foundation of your map or scene; it provides context for your data and defines how your map or scene looks. Choosing the right basemap can:

- Align your map or scene with your brand
- Help tell your story
- Make your map layers and graphics easier to see and understand
- Make your app feel like a part of your user's Geographic Information System (GIS) when your app connects to a portal

Copyright © 1995-2020 Esri. All rights reserved.

There are many attractive basemaps available on ArcGIS Online. If you're creating an editing or browsing experience, your users may be accustomed to having certain basemaps available through their portal. You can display a list of featured basemaps; see Get a list of featured basemaps.

It is also possible to publish your own basemaps and customize the style of the existing vector basemaps. See How to Customize Esri Vector Basemaps.

Time-aware layers

Although all layers support geographic information, time-aware layers display time information as well. When a layer supports time, you can:

- Filter layer content based on time
- Compare data over time using time offsets

Copyright © 1995-2020 Esri. All rights reserved.

For more information, see Visualize and compare data over time.

Image adjustment layers

Image adjustment layers let you change brightness, contrast, and gamma for imagery. See Layer types described for more information.

Layer performance considerations

Vector & raster data

There are two main formats for the display and management of geographic data:

- **Vector** — data is described by points, lines, and polygons (vectors). The client renders the vector content into raster images for display on the device.
- **Raster** — data is transmitted via pre-rendered images; the client doesn't have information about the underlying features that were rendered into the image.

Because vector layers are rendered on device, they can always be perfectly suited to the size and resolution of the client display. Vector data is also very small, making it ideal for taking offline or use in low-bandwidth environments. Vectors are not suitable for representing data that comes from raster sources, like satellite imagery.

Some basemaps combine both raster and vector data. For example, the world imagery with streets (vector) basemap from ArcGIS Online uses a tiled raster layer for imagery and a vector layer for features and labels. This ensures that features and labels are always crisp while still showing raster imagery.

Dynamic and tiled layers

Services can take one of two approaches to providing their data:

- **Dynamic** — Regenerate a map or scene for each request based on information provided by the client, like desired image size and extent.
- **Tiled** — Generate tiles ahead of time, then serve them as needed by the client.

Because dynamic maps and scenes are generated as needed, they are more appropriate for maps or scenes with data that changes often. The downside to dynamic maps or scenes is that they require more server resources – rendering map images is an expensive operation. For things that don't change often, tiled services are more appropriate.

Copyright © 1995-2020 Esri. All rights reserved.

Tables

Tables provide a source of data that can be worked with directly, or perhaps displayed in a feature layer. Tables in ArcGIS define a schema for features – a consistent set of fields that features are expected to have. Tables can have many features. Entries in spatial tables can have geometries. Tables can come from several sources:

- Shapefiles
- Geopackages
- Maps or scene
- ArcGIS feature services
- WFS feature services
- Geodatabases

Tables can be:

- Queried
- Edited (depending on the source)
- Shown in a feature layer
- Taken offline

Non-spatial tables

Non-spatial tables are just like other tables, except they don't have a geometry column. Because they don't have geometry, they can't be rendered in a layer. Although non-spatial tables can't be rendered, they can still be added to maps or scenes as data sources for querying. Features (entries) in spatial tables can be related to the features in non-spatial tables. (See Relate features for information about associating features from different tables.) Non-spatial tables are represented by the same Runtime types as spatial tables.

Table performance concepts

A table might have hundreds of thousands of features. How your app requests features from the table is a performance-critical decision.

When working with tables that come from an ArcGIS feature service or WFS (rather than a local file), **feature request modes** control how often features are retrieved and how they are cached. There are three feature request modes:

- **On interaction cache** (default) — This mode is best for scenarios involving services with large amounts of infrequently edited data. This mode is recommended if you have an occasional or unreliable connection to the service. When the table is used in a layer in a map or scene, features are requested for the visible extent (map area) and are cached locally. Features continue to be requested and cached as you navigate the map or scene to other areas. Cached features are not requested again. Features can also be requested by querying or populating the table directly. Runtime chooses the optimal way to execute a query, either querying the local cache, or querying the service and importing the results into the local cache. For example, if a spatial query is executed for the current map extent, results are obtained from the local cache—this avoids re-fetching features from the service.
- **Manual cache** — This mode is recommended when the data from the service is not expected to change frequently and when it is not critical for the client app to always have the most recent changes. The amount of data populated to the local cache from the service is expected to be small. The table's local cache can only be populated by calling

Copyright © 1995-2020 Esri. All rights reserved.

`PopulateFromServiceAsync`. Features are never requested automatically, and the cache is never populated based on map interaction.

- **On interaction, no cache** — This mode is recommended if it is critical for your app to always be working with the latest data. This mode is not recommended if you have an occasional or unreliable connection to the service. When the table is used in a layer in a map or scene, features are requested for the visible extent and are not cached locally. Features are requested from the server each time you navigate the map, regardless of whether the extent has been visited before. Queries are always executed against the server. Navigating the layer or querying the table both cause the local cache to be cleared and repopulated. Because all operations involve querying the server, an always-on service connection is required. Anticipate higher data usage than the other interaction modes.

Note:

WFS feature tables only support **manual cache** feature request mode. **On interaction cache** is the default so WFS feature tables must be set to use **manual cache** before attempting to load features.

Copyright © 1995-2020 Esri. All rights reserved.

Layer types described

This topic provides the following:

- A bird's-eye comparison of layers (table format)
- A detailed description of layers

For more information on choosing the right layer types for your app, see Layers and tables. If you're looking for a quick way to add points or text to a map, consider using a graphics overlay.

Layer type	2D/ 3D	Time-aware	Online/ offline	Sources
Annotation layer	2D	No	Both	ArcGIS feature service, portal item, mobile map package (.mmpk)
ArcGIS map image layer	Both	Yes	Online	ArcGIS map service
ArcGIS tiled layer	Both	No	Both	ArcGIS tile service, tile package (.tpk/.tpkx)
ArcGIS vector tiled layer	2D	No	Both	ArcGIS vector tile service, vector tile package (.vtpk)
Bing maps layer	Both	No	Online	Bing maps
ENC layer	2D	No	Offline	ENC exchange set, ENC cell
Feature collection layer	Both	No	Both	Portal item, web map, feature set / query result
Feature layer	Both	Yes	Both	ArcGIS feature service, WFS, shapefile, GeoPackage, geodatabase
Group layer	Both	No	Both	Other layers and group layers
KML layer	Both	Yes	Both	KML file (.kml, .kmz)
Mobile basemap layer	2D	No	Offline	Mobile map package (.mmpk)
OpenStreetMap layer	Both	No	Online	OpenStreetMap.org
Raster layer	Both	Yes	Both	GeoPackage, raster file, ArcGIS Image service
Scene layer	3D	No	Both	Scene service or scene package
Subtype feature layer	2D	No	Both	ArcGIS feature service, geodatabase
Web tiled layer	Both	No	Online	Web tile service
WMS layer	Both	Yes	Online	WMS service
WMTS layer	Both	No	Online	WMTS service

There is also a Runtime layer types overview table with more information about layer data formats, rendering schemes, and feature support.

Annotation layer

API types: `AnnotationLayer`, `AnnotationSublayer`

Copyright © 1995-2020 Esri. All rights reserved.

Annotation can be used to symbolize text on your maps (in addition to labeling). Annotation is defined by a text string, geographical location, and display properties including font, size, and color. These are stored together in an annotation feature class within a geodatabase.

Functional characteristics

You can create an annotation layer by providing the URL or portal item of an annotation feature class or from a feature table containing annotation. Annotation layers are also supported in mobile map packages created in ArcGIS Pro 2.3 or later. Currently, ArcGIS Runtime supports read-only standard annotation.

An annotation layer may contain annotation sublayers (known as annotation classes in ArcGIS Pro). Annotation sublayers offer finer control over your annotation, allowing the data's author to set different visual properties from the parent annotation layer. These visual properties include font, size, color, or different minimum and maximum scale ranges. Within ArcGIS Runtime, the annotation sublayer allows you to do the following:

- Access the annotation sublayer's metadata information, including legend information.
- Set the visibility of each annotation sublayer.

Performance characteristics

Annotation respects the reference scale defined by the map, so annotation will always be presented to the user at the correct size and position, as defined by the annotation author.

Learn more

- Sample: Display annotation
- Sample: Control annotation sublayer visibility

ArcGIS map image layer

API types: `ArcGISMapImageLayer`, `ArcGISMapImageSublayer`

ArcGIS map image layer displays maps from an ArcGIS map server. A map service can contain multiple layers, rendered by the server each time a request is made and returned to the client as a single raster image. While the image itself does not contain information about the features it displays, you can get the underlying data (service feature table) for each sublayer. This layer supports time-based filtering.

Functional characteristics

Map images are created and returned by the server on every request, so they always show the latest data at the time of the request. Characteristics of the image, such as brightness, contrast, gamma, and opacity can be specified. You can also control the visibility and symbols of sublayers and filter data with layer definition expressions. The spatial reference can be changed from the service's default and the service will reproject each image on the fly.

The underlying service feature table for each map image sublayer (or for nonspatial tables used in the service) can be accessed. These tables can be queried using any valid combination of attribute, spatial, and temporal criteria. You can also query for summary statistics or to find related features in other tables.

Copyright © 1995-2020 Esri. All rights reserved.

Performance characteristics

The map service creates map images on the fly. Rendering time depends on the amount and complexity of the data in the map. This will typically be slower than fetching the equivalent map as prerendered tiles. Because the server renders the map, map image layers require less processing time on the client than similar maps rendered locally.

ArcGIS map image layers are good candidates for showing features that change periodically over time, or that require some sort of filtering by the user. Although rendering occurs on the server, the client has access to service feature tables for all sublayers (as well as nonspatial tables and relationships).

Learn more

- Sample: ArcGIS map image layer (URL)
- Sample: Change sublayer visibility
- Guide: What is a map service?
- Guide: Map authoring considerations

ArcGIS tiled layer

API types: `ArcGISTiledLayer`, `TileCache`

ArcGIS tiled layers consume raster tiles provided by an ArcGIS service or a tile package. Raster tiles are cached by the server at various scales; the client requests the tiles needed at a particular map extent. You can use `ExportTileCacheTask` to generate and download tiles from the service, creating a tile package (`.tpk`/`.tpkx`). Alternatively, you can use ArcGIS Pro to create a map tile package and provision it to the device. See Take a layer offline for more information.

Functional characteristics

ArcGIS tiled layers do not support reprojection, query, select, identify, or editing.

Performance characteristics

Tiles are generated by the server only once when the service is originally created. Requests for tiles are made on multiple threads and handled asynchronously. The size of each returned tile increases as the resolution or complexity of the image in the tile increases. For example, high-resolution imagery tiles can be larger in file size than topographic mapping for the same area and map extent.

Learn more

- Sample: ArcGIS tiled layer (URL)

ArcGIS vector tiled layer

API types: `ArcGISVectorTiledLayer`, `VectorTileCache`

An ArcGIS vector tile layer consumes vector tiles and an associated style for drawing them. There are many styled vector basemaps available through ArcGIS online; if you can't find one that works, it is easy to create your own with the online style editor. Your customized vector layers can be saved to and read from ArcGIS Online.

Copyright © 1995-2020 Esri. All rights reserved.

You can use `ExportVectorTilesTask` to generate and download a vector tile cache (optionally including a custom style) directly to your device. The cache will be a single-vector tile package (`.vtpk`) file. Alternatively, you can use ArcGIS Pro to create a vector tile package and provision it to the device. See Take a layer offline for more information. Or, you can use ArcGIS Pro to create a mobile map package that contains a vector tile package and your custom style, if required.

Functional characteristics

Vector tiled layers do not support identify, query, selection, or time. Vector tiled layers cannot be displayed in scenes.

Performance characteristics

Because vector tiles are defined in terms of points, lines, and polygons, they are very fast to draw. Vector data is usually smaller than raster data and may be more suitable in low-bandwidth or offline environments. Because vector tiles are rendered on the device, they always look good, regardless of screen size or resolution.

Learn more

- Sample: ArcGIS vector tiled layer (URL)
- Story map: Customizing Esri vector basemaps
- Story map: Customizing Esri vector tiles with sprites and fonts

Bing maps layer

API types: `BingMapsLayer`

Bing maps layer is a layer for showing tiled raster basemaps from Bing.

Functional characteristics

Bing maps require an API key from Microsoft. A Bing maps key may be provided via a portal that has one, or with a string at load time. See Microsoft's documentation for details on getting a key.

Bing maps layers do not support identify, query, selection, or time.

Performance characteristics

Bing maps layers perform similarly to other tiled layers. Raster tiles are fetched from Bing's servers on demand.

Copyright © 1995-2020 Esri. All rights reserved.

Learn more

- Guide: Bing maps API keys in Portal
- Microsoft: Getting a Bing maps API key

ENC layer

API types: `EncLayer`, `EncCell`, `EncDataset`

Electronic navigational charts (ENCs) are georeferenced vector datasets for the visualization and analysis of hydrographic and maritime information. ArcGIS Runtime supports ENCs that conform to the International Hydrographic Organization (IHO) S-57 standard. For more information, see Display electronic navigational charts.

Functional characteristics

ENC layer supports identify and selection. ENC layer does not support query or time. The ArcGIS Runtime ENC implementation is not suitable for use in navigation or the creation of an Electronic Chart Display and Information System (ECDIS).

Performance characteristics

ENC layers differ from other layer types in that ENC content is cached in an internal collection of SENC files. When an ENC layer is displayed, only the content in the SENC files is referenced. SENC files are produced when an ENC cell is read for the first time and updated as update sets are applied. You can set the path to the SENC storage directory with the `EncEnvironmentSettings.SencDataPath` property. SENC files are in a binary format that is not intended to be shared between platforms or versions of ArcGIS Runtime.

Learn more

- Guide: Display electronic navigational charts
- Sample: Add an ENC exchange set
- Sample: Change ENC display settings
- Sample: Select ENC features

Feature collection layer

API types: `FeatureCollectionLayer`

A feature collection layer is used to display the features in a feature collection. A feature collection provides a way of grouping logically related feature collection tables. Tables in the collection can have different schema, geometry types, and rendering. A feature collection layer allows multiple tables with different schemas to be managed as a group.

Functional characteristics

Redlining information (called Map Notes in ArcGIS Online), may contain points, lines, polygons, and associated text to describe things in the map. Because they have different schema and geometry types, these features are stored in several distinct tables. A feature collection layer allows these tables to be rendered and managed as a group.

A feature collection can be saved in the map or as a stand-alone portal item. If you need to share the feature collection between several maps, it's best to store it as a separate portal item. If you need to make frequent (near real-time) edits to features in a collection, consider storing these in a feature service instead, because the feature collection is not refreshed until

Copyright © 1995-2020 Esri. All rights reserved.

the map or portal item is reloaded. If features are used by a single map or are not subject to frequent updates, it might be best to store them directly in the map. Edits made to features stored in a map will be saved when the map is saved. Edits made to features stored in a portal item (and loaded into a map, for example) must be explicitly saved to the original portal item.

Performance characteristics

Feature collection layers are designed to display a moderate amount of feature data (hundreds or thousands of features). They are ideal for sharing static data (not subject to frequent updates, in other words) among several clients.

As full feature information is cached locally in a geodatabase and features are drawn natively, this layer type offers excellent display performance when zooming and panning the map, within the extent of cached features.

Downloading features to the device may require extensive network usage and local device storage space. Once the features are present on the client, the app no longer needs to make requests for the data. App memory increases with the number and complexity of the features in the collection.

Learn more

- Sample: Create feature collection layer
- Sample: Feature collection layer from portal
- Sample: Feature collection layer from query

Feature layer

API types: `FeatureLayer`, `ServiceFeatureTable`, `GeodatabaseFeatureTable`, `ShapefileFeatureTable`, `GeoPackageFeatureTable, WfsFeatureTable`

Feature layers display data from feature services or supported local data sources, including shapefiles, GeoPackages, and geodatabases. Feature layers can be used to display, select, and query features in a layer. If the underlying feature service or table supports editing, you can use it with a feature layer as a basis for editing geometry, attributes, and attachments. Feature layers are used to store features associated with a utility network.

Features are retrieved as needed by the app. Features can be downloaded from a sync-enabled feature service when the device is connected and cached locally for use when the device is offline. Edits can then be synchronized back to the service.

Functional characteristics

The following are sources that can be rendered using a feature layer:

- Feature service—Backed by a service feature table; feature data from the service is cached locally in the table. New features are retrieved automatically when you navigate the map. The local table cache is discarded when the layer is disposed. If sync is enabled, features can be created, edited, and pushed to the server.
- Geodatabase—Backed by a geodatabase feature table; The geodatabase can be a replica of a feature service, which allows synchronizing with a feature service, or taking the content of a feature service offline. Use a geodatabase sync task to synchronize the geodatabase with the service.
- Shapefile—Backed by a shapefile feature table; use a feature layer to show the contents of shapefiles (`.shp`).
- Geopackage—Backed by a geopackage feature table; use a feature layer to render the tables in a GeoPackage (`.gpkg`). A GeoPackage is a data source that conforms to the OGC GeoPackage specification. Geopackage feature tables can be edited

Copyright © 1995-2020 Esri. All rights reserved.

and saved, but can't support sync, because there is no backing feature service. ArcGIS Runtime supports GeoPackage versions 1.0, 1.1, and 1.2.

- Web Feature Service (WFS)—Backed by a WFS feature table. You can populate the table using Runtime query parameters or raw XML-encoded GetFeature queries. A WFS feature table only supports manual cache feature request mode. See Performance characteristics for more details. Runtime supports OGC WFS versions 2.0.0 and 2.0.2.
  - WFS server implementations are inconsistent in how they expect coordinates to be formatted. Some return and expect coordinates in (x,y) order, while others expect (y,x). Runtime guesses the correct order by default. This behavior can be configured with the `AxisOrder` and `FilterAxisOrder` properties.

Individual features can be queried and filtered based on spatial queries or SQL queries. Introduced at 100.3.0, string comparisons for features queried in service feature tables are case insensitive.

Local tables cannot be reprojected automatically. Find out more about editing features.

For details on viewing, identifying, querying, and editing features offline, see Take a layer offline for more information.

Feature layers expose an `unSupportedJson` property that, for feature layers based on a feature service, returns a dictionary of values known to the supported web map specification but not explicitly exposed through the Runtime API. This allows you to access information that was saved with the layer but not used by Runtime. Feature layers also provide an `unKnownJson` property to return JSON that was not recognized.

Performance characteristics

As full feature information is cached locally in a geodatabase, shapefile, or GeoPackage, and features are drawn natively, this layer type offers excellent display performance when zooming and panning the map within the extent of cached features. Querying features is also efficient, enabling app functions such as real-time updates of query results in a map.

The local cache must be initially created, which can be resource-intensive for the server. The initial download to the device may require extensive network usage and subsequent local device storage. App memory increases with the number and complexity of the features cached. Network usage can be eliminated by provisioning the cache directly to the device in advance.

Feature tables backed by a service define three feature request modes. The table's feature request mode controls how and when features are requested from the service:

- On interaction cache——Features are requested automatically for the visible map or scene extent. As the user pans and zooms, features are cached locally. If the user returns to an area where features have already been loaded, the table won't need to download those features again.
- Manual cache—Features must be manually populated using a call to `PopulateFromServiceAsync`. Once populated, all queries are made against the local table only. `PopulateFromService` can be called again to retrieve more features from the service.
- On interaction, no cache—Features are requested automatically for the visible map extent. As the user pans and zooms, features outside the visible extent are not cached and must be downloaded again each time.

See Layers and tables for more information about feature request modes.

Learn more

- Sample: Feature layer (geopackage)
- Sample: Feature layer (query)
- Sample: Feature layer (shapefile)

Copyright © 1995-2020 Esri. All rights reserved.

- Sample: Service feature table (manual cache)
- Guide: Symbols, styles, and renderers
- Guide: Edit features

Group layer

API types: `GroupLayer`

A group layer is a dynamic collection of operational layers and other group layers. It is designed for presentation of operational layers that are related by theme. You can add several related operational child layers into one group layer so the layers can be displayed together. Suppose there are several feature layers that represent existing or new buildings, sidewalks, roads, trees, and other infrastructure in a development project. You can add the feature layers for existing features to a single group later called Existing, and add the planned features into another group layer called Planned. Then, you can manage the visibility of the existing or planned features as separate groups.

Any operational layer can be added to a group layer. That is, any operational layer that can be added to a map or scene could be added to a group layer in that map or scene instead.

Functional characteristics

Group layers function as do other layers:

- Each group layer has visual properties (visibility, opacity) that can be applied to all layers in the group layer at once. This is helpful when controlling the visibility of the group.
- You can set a scale range on a group layer using the minimum and maximum scale properties. If a child layer has more restricted scale range, that scale range will be honored for that child layer.
- A group layer may be queried for the aggregate geographic extent of its child layers.
- A group layer may be nested inside another group layer. There is no defined nesting level limit.
- Cloning a group layer will also clone its child layers.
- A group layer doesn't have its own attribution text. The text from the child layers is displayed instead.
- A group layer is not time-aware, but may contain layers that are.
- Each of the child layers can have a spatial reference. However, a group layer has no spatial reference of its own.
- A group layer cannot be part of any basemap.

When identifying layers in a map or scene, identification will operate on all the child layers of any group layers. See Identify on group layers.

You can query a group layer for its name and the name and symbol or symbols used for each child layer. This is useful information for building a legend. The `ILayerContent` interface gives you access to any names or symbols relevant for a layer. `ILayerContent.GetLegendInfosAsync` returns a list of `LegendInfo` with the name and symbol for each symbol used to render the layer. For a group layer, this consists of a single `LegendInfo` with the layer name and a null symbol. To get a group layer's child layers, the `ILayerContent.SubLayerContents` property returns a list of `ILayerContent`, one for each child layer.

The following example shows a function that can recursively read the contents of a layer (including a group layer) to create a legend:

```
// Create a list to hold legend information (layer names, symbol swatches).
// In this example, "MyLegendItem" is a custom data structure with Name and SymbolSwatch
```

Copyright © 1995-2020 Esri. All rights reserved.

```
properties.
private IList<MyLegendItem> legendInfoList;

// Use a function that can recursively iterate a group layer (ILayerContent) and add to the
legend list.
private async Task ReadLegendInfoRecursiveAsync(ILayerContent content)
{
    // Make sure the ILayerContent is valid and should be shown in the legend.
    if (content == null || !content.ShowInLegend) { return; }

    // Get the legend info for the content (symbols, for example).
    IReadOnlyList<LegendInfo> layerLegendInfos = await content.GetLegendInfosAsync();

    // If legend info exists, add each legend info item to the list.
    if (layerLegendInfos != null)
    {
        // If more than one info (symbol), add the layer name before the symbol legend info.
        if(layerLegendInfos.Count > 1)
        {
            // Create a legend item with just the layer name (no symbol).
            MyLegendItem layerItem = new MyLegendItem(content.Name, null);
            legendInfoList.Add(layerItem);
        }

        // Iterate all legend items (symbols).
        foreach (LegendInfo itemInfo in layerLegendInfos)
        {
            // Get the symbol name. If there isn't one, use the layer name (probably a single
symbol renderer).
            string symbolName = !string.IsNullOrEmpty(itemInfo.Name) ? itemInfo.Name :
content.Name;

            // Get the symbol swatch image (if there is one).
            ImageSource imageSource = null;
            if (itemInfo.Symbol != null)
            {
                // Get the symbol swatch image source.
                RuntimeImage swatch = await itemInfo.Symbol.CreateSwatchAsync();
                imageSource = await swatch.ToImageSourceAsync();
            }

            // Add an item to the legend list for the symbol.
            MyLegendItem symbolItem = new MyLegendItem(symbolName, imageSource);
            legendInfoList.Add(symbolItem);
        }
    }

    // Add a blank legend item (to create space).
    MyLegendItem blankItem = new MyLegendItem("", null);
    legendInfoList.Add(blankItem);

    if (content.SublayerContents != null)
    {
        // Recursively call this function again for each sublayer.
        foreach (ILayerContent subLayer in content.SublayerContents)
        {
            await ReadLegendInfoRecursiveAsync(subLayer);
        }
    }
}

// Use code like the following to create a new list populate it with a group layer legend info.
legendInfoList = new List<MyLegendItem>();
```

Copyright © 1995-2020 Esri. All rights reserved.

```
await ReadLegendInfoRecursiveAsync(groupLayer);

// Show the list in a list box control to display the legend.
LegendListBox.ItemsSource = legendInfoList;
```

Tip:

A Legend control that implements code such as the previous example is available in the ArcGIS Runtime SDK for .NET Toolkit.

All child layers can be omitted from the legend using `GroupLayer.ShowChildrenInLegend`.

Performance characteristics

Full extent is derived asynchronously based on the information that is available from the child layers. This means the full extent can change when child layers are added or removed. The full extent geometry will have the spatial reference of the first loaded child layer.

To zoom to the full extent of a group layer, first interrogate each of the child layers' loaded states to determine if the full extent of the group layer is representative of all layers. Each child layer must be loaded for its extent to be included in the group layer full extent.

Group layers are not supported in web maps. Saving a map with a group layer to a web map flattens the operational layers. Group layers are supported in web scenes.

The visual opacity of child layers will be the mathematical product of its opacity and the group layer's opacity. Opacity values range from `0.0` (transparent) to `1.0` (opaque). This means that the opacity property of the group layer affects the opacity of the child layers wherein `0.0` will make all child layers transparent, and `1.0` will set the opacity of the child layers to their by-layer opacity.

KML layer

API types: `KmlLayer`, `KmlDataset`

KML is a specification for working with geographic content in Google Earth. KML files can contain 2D and 3D content, as well as links to content from the network.

Functional characteristics

KML represents features as a tree of nodes, including network links, folders, and placemarks. Like ArcGIS features, placemarks are defined with point, line, or polygon geometry. KML geometry, however, is always based on the WGS84 coordinate system. KML can also have attributes, but unlike ArcGIS feature services, a KML document does not use a schema to define a standard

Copyright © 1995-2020 Esri. All rights reserved.

set of fields. ArcGIS Runtime supports version 2.2 of the KML specification as defined by the Open Geospatial Consortium (OGC). See Display KML content for more details on displaying KML data. Creating, editing, and saving KML data in ArcGIS Runtime is described in more detail in the Edit KML content topic.

Unique features of KML include network links and screen overlays. Network links are useful for retrieving network content on a specified interval—perfect for keeping a live map up to date. Network links can also be used to split a large KML file into multiple per-region KML files, with each being loaded only when the region is in view. Screen overlays can be used to show content, such as branding or a legend, on top of the view. Screen overlay content does not scroll with the map or react to user interaction.

Demonstration of how screen overlays are rendered on top of the map, staying at a fixed position on the screen as the user pans and zooms the map.

Performance characteristics

KML files can vary significantly in the features they use, including 3D models, network links, and refresh intervals. Not all features are supported in 2D and 3D. For example, 3D models only appear when the KML layer is shown in a scene. Many KML files consist solely of a pointer to another KML file and a refresh interval. For example, the National Weather Service distributes forecast maps in the United States this way.

Learn more

- Guide: KML in ArcGIS Runtime
- Guide: Edit KML content

Mobile basemap layer

API types: `MobileBasemapLayer`

A mobile map package can be created in ArcGIS Pro and provisioned for display in your app. The package may contain one or more mobile maps, each of which can contain a basemap with mobile basemap layers. These layers contain vector features in a compressed format. Because mobile basemap layers are authored in Pro and can't be used independently of a mobile map package, there is no mobile basemap layer constructor.

Copyright © 1995-2020 Esri. All rights reserved.

Functional characteristics

Mobile basemap layers do not support query, selection, identify, or time. Data must be prepared using ArcGIS Pro as part of the mobile map package (`.mmpk`) authoring process.

Performance characteristics

Mobile basemap layers contain vector content that has been packaged specifically for use offline. Tiles are loaded locally, which means they can be displayed quickly and don't require network access.

OpenStreetMap layer

API types: `OpenStreetMapLayer`

OpenStreetMap layer is a preconfigured web tiled layer that displays tiles from the openstreetmap.org tile servers. OpenStreetMap (OSM) is a project to create a free, publicly editable map of the world.

Functional characteristics

OpenStreetMap layer is a preconfigured web tiled layer. Unlike a web tiled layer, the OSM layer comes with predefined attribution. OpenStreetMap layers do not support identify, query, selection, or time.

Note:
OpenStreetMap is free for everyone; OpenStreetMap tile servers are not.

Use of OpenStreetMap's tiles is subject to their tile usage policy. Consider the following before using OpenStreetMap layer in your app:

- Service availability is not guaranteed; apps with high demand may be denied access at any time.
- There are many organizations providing OpenStreetMap tiles; consider using one of these services (or hosting your own) if you anticipate high demand.

Performance characteristics

Performance of OSM layers is similar to web tiled layer performance. Raster tiles are read from the service as needed by the map. Tiles may be rendered on demand at high zoom levels.

Learn more

- About: OpenStreetMap.org
- Policy: OpenStreetMap tile usage policy
- Alternative tile server: Stamen (use with web tiled layer)

Raster layer

API types: `RasterLayer`

Raster layer displays data from a raster data source. Data sources include the following:

- Raster files

Copyright © 1995-2020 Esri. All rights reserved.

- Mosaic datasets
- ArcGIS Image services
- Raster functions

You can change how a raster layer is visualized by creating a raster renderer and applying it to the layer. For more information about working with rasters, see Add raster data.

Functional characteristics

Raster imagery is often used as part of a basemap. Many sources of analytical data are available in a raster format, such as those derived from satellite imagery. This data may be used for classification, change analysis, and so on. Raster layers support identify.

A raster consists of a matrix of cells (or pixels) organized into rows and columns (or a grid) where each cell contains a numeric value. The values in a raster can represent discrete features, such as land use codes, or continuous phenomenon such as elevation.

An example of a single-band raster dataset is a digital elevation model (DEM), where each cell contains one value representing the elevation at that location. A satellite image, on the other hand, commonly has multiple bands representing different wavelengths of the electromagnetic spectrum. Landsat imagery, for example, contains seven bands that represent data from the visible and infrared parts of the spectrum. Rasters are particularly useful for remote sensing tasks, such as monitoring vegetation and seeing through smoke to analyze an active fire.

Performance characteristics

Raster datasets can be quite large. The size of the dataset depends on the following:

- The geographic extent of the data
- The size of the cells used (resolution)
- The number of bands

As with any data consumed locally, file size can be an issue for storage as well as for transferring datasets over the network.

Learn more

- Sample: Raster hillshade renderer
- Sample: Raster layer (file)
- Sample: Raster layer (geopackage)
- Sample: Raster layer (service)
- Sample: Raster RGB renderer
- Sample: Raster stretch renderer
- Guide: NDVI function

Copyright © 1995-2020 Esri. All rights reserved.

Scene layer

API types: `ArcGISSceneLayer`

Scene layers display content from a scene service or a scene package. Scene layers can be used to show 3D objects, such as textured or untextured buildings, or 3D mesh data, such as imagery captured by drones.

Note:
Many layer types work in 3D, not just scene layer.

Content in 3D can be delivered as scene packages (stored on the client) or as online scene services. These data sources support consuming large volumes of multipatch features, such as building models for an entire city. See Build a new scene for more information about working with scenes and displaying data in 3D.

Functional characteristics

A scene service is used to provide one of the following types of 3D data:

- 3D objects—3D object scene layers contain point features with associated 3D models used to represent objects such as buildings, trees, and street furniture that are explicitly modeled in three dimensions. These features have attributes and can be identified.
- Integrated mesh—3D mesh data is typically captured by an automated process for constructing 3D objects out of large sets of overlapping imagery. The result integrates the original input image information as a textured mesh including 3D objects, such as buildings and trees, and elevation information. Integrated mesh scene layers are often created for citywide 3D mapping. They can represent something as small as a single building or as large as a city or state. They are often assembled automatically from imagery collected by a drone; they cannot be restyled because there is no underlying feature data.
- Point scene—To ensure fast visualization, cached scene layers are used to display large amounts of 3D point data about individual features, such as all the trees in a city. Point scene layers have features that can be identified and that have attributes that may be cached or available from an associated feature layer. The display of point scene layers is automatically thinned to improve performance and visibility at smaller scales and longer distances. Automatic thinning means that not all features are displayed at small scales; as you zoom in, additional features are displayed.
- Point cloud—Point cloud layers provide fast display of large volumes of symbolized and filtered point cloud data. Point clouds are optimized for the display and sharing of data from many kinds of sensors that create point clouds, such as lidar.

Point scene layers are generated from point feature layers. Point scene layers can be generated by ArcGIS Pro and hosted as a scene service or scene layer package.

Note:
When publishing a scene from ArcGIS Pro (sharing as a web scene) any point feature layers in the **3D Layers** section of the **Contents** pane will be published as a scene service. The scene service will be referenced within the web scene.

Performance characteristics

For a scene layer, the rendering of 3D data is based on the current level of detail (LOD), which is determined by the distance from the camera. Each object in a 3D object service is represented differently at various LODs determined by the camera distance. As the camera distance changes in the scene, objects will be displayed appropriately for the current LOD. This results in good client performance and low memory usage, at the expense of frequently loading and unloading objects.

Copyright © 1995-2020 Esri. All rights reserved.

Learn more

- Sample: Scene layer (URL)
- Guide: Multipatches

Subtype Feature Layer

API types: `SubtypeFeatureLayer, SubtypeSublayer`

Subtypes provide a way to organize your data within an ArcGIS Feature table. For example, the streets in a city streets feature table could be categorized into three subtypes: local streets, collector streets, and arterial streets. The benefit of the subtype feature layer is that it allows you to configure different layer properties per subtype.

This is particularly useful for data, such as utility networks, in which many different network elements are grouped into a single subtype feature layer.

Functional characteristics

The subtype feature layer acts like a group layer that automatically creates one feature layer, or subtype sublayer, for each subtype in the table.

Each subtype sublayer can be configured independently of one another. They can have different renderers, label definitions, max and min scales, visibility, layer order in the map, and so on.

Performance characteristics

Queries are optimized across all sublayers. For example, a pan of the map sends a single request to the feature table to cover all the sublayers. If you had manually created a feature layer for each subtype, this would have resulted in one query per feature layer.

Learn more

- Introduction to subtypes.

Web tiled layer

API types: `WebTiledLayer`

Web tiled layer displays tiles from a tile service. Tiles are accessed directly by URL using a defined URL template. Because the service is only providing images, the developer is responsible for manually setting the attribution text on the map or scene view.

```
string _templateUri = "http://{subDomain}}.tile.st.com/terrain/{level}/{col}/{row}.png";
List<string> _tiledLayerSubdomains = new List<string> { "a", "b", "c", "d" };
WebTiledLayer myBaseLayer = new WebTiledLayer(_templateUri, _tiledLayerSubdomains);
```

Functional characteristics

Tiles are fetched on demand using the provided URL template. Tiles are typically prerendered (cached) on the server but may be generated on demand by some services. Web tiled layers do not support identify, query, selection, or time.

You can specify subdomains from which the layer will request tiles. This allows the load to be evenly distributed among servers.

Copyright © 1995-2020 Esri. All rights reserved.

Performance characteristics

Web tiled layer requires a connection to the service at all times. Performance is similar to other raster tile layers.

Learn more

- Sample: Web tiled layer

Web Map Service (WMS) layer

API types: `WmsLayer`, `WmsSublayer`, `WmsService`

WMS layer displays data from a web service that provides maps in the form of server-rendered images. Web Map Service (WMS) is an Open Geospatial Consortium (OGC) standard for delivering map images from an online service. ArcGIS Runtime supports WMS versions 1.1.0, 1.1.1, and 1.3.0.

Functional characteristics

A WMS service can contain multiple layers in a hierarchy. A `WmsLayer` can be constructed directly with a URL to a service and the uniquely identifying name of the desired layer. Alternatively, a `WmsService` can be used to programmatically explore the available layers and allow the user to choose layers at run time.

The maps provided by a WMS service use predefined symbology defined by the server. As a result, it is not possible to apply custom renderers or to visualize feature selection. WMS layers can have multiple style options. Your app can choose from the available styles.

Some layers can be marked as opaque, which means that they cover most of the map area and are good candidates for use as a basemap.

WMS layers support identify and time. They do not support selection or query. Note that due to the nature of WMS, it is not possible to retrieve feature geometry from WMS identify results. WMS layers support custom parameters, which can be specified for the service or an individual layer.

Performance characteristics

WMS servers render map images on demand, which can require more server resources than a similar tiled service. WMS requires a service connection at all times.

Learn more

- Sample: Add a WMS layer
- Sample: WMS service catalog
- Sample: Identify WMS features
- Specification: OGC WMS standard

Web Map Tile Service (WMTS) layer

API types: `WmtsLayer`

WMTS layers display map tiles from a WMTS service. Web Map Tile Service is an Open Geospatial Consortium (OGC) standard for delivering geographic data via raster tiles. ArcGIS Runtime supports WMTS 1.0.0.

Copyright © 1995-2020 Esri. All rights reserved.

Functional characteristics

The maps provided by a WMTS service use predefined symbology defined by the server. As a result, it is not possible to apply custom renderers or to visualize feature selection.

A WMTS service can contain multiple layers in a hierarchy. A WMTS layer can be constructed directly with a URL to a service and the uniquely identifying name of the desired layer. Alternatively, a WMTS service can be used to programmatically explore the available layers and allow the user to choose layers at run time.

Performance characteristics

WMTS layer consumes raster tiles that were prerendered by a server. WMTS requires fewer server resources than WMS because the images are rendered and cached ahead of time. WMTS layer requires a connection to the service at all times.

Learn more

- Sample: Add a WMTS layer
- Specification: OGC WMTS standard

Copyright © 1995-2020 Esri. All rights reserved.

Features and graphics

In ArcGIS Runtime, features and graphics represent real-world objects on a map or scene. Every feature and graphic has a geometry representing its shape and location, as well as other attributes that further describe the represented object. For example, polygon features could represent land parcels and include attributes about each one such as parcel ID or owner. Point graphics could represent event locations, including the event's time and type.

Features and graphics are similar to one another in that they both have a geometry and attributes, this is represented by the geoelement base class which they both implement. However, features are designed for different uses than graphics. Your choice between using a graphic or a feature depends on various factors, such as whether they are persisted in a data store or a map, whether they all share a single geometry type and set of attributes, or how your app displays them. This topic describes and compares features and graphics and helps you choose which is best to use in various cases.

The following table compares important characteristics of features and graphics:

Comparison of features and graphics

Characteristic	Feature	Graphic
Display method	In a feature layer in a map in a map view or scene in a scene view.	In a graphics overlay in a map view or scene view.
Persistence	In a feature table in a data store (such as a database or service) or in a map or scene.	In app memory only.
Geometry type (point, line, and so on)	Features of different geometry types **cannot** exist in the same layer.	Graphics of different geometry types **can** exist in the same graphic overlay.
Attributes	Features in the same data store or feature layer have a common attribute schema.	Each graphic may have a set of attributes unlike other graphics in the same graphic overlay.
Symbolization	Symbolized according to the renderer defined for the feature service or the feature layer. If persisted in a feature collection table in a map or scene, the renderer can be overridden with feature-specific symbols.	Symbolized individually or according to the graphics overlay's renderer.
Identifiable	Via the map view or scene view.	Via the map view or scene view.

Copyright © 1995-2020 Esri. All rights reserved.

Characteristic	Feature	Graphic
Editable	Created, updated and deleted using feature table operations	Created through a graphic constructor and added to the graphics overlay graphics collection. Attributes and geometry updated by reference on a graphic instance

Symbolizing features and graphics

In order to display on a map or scene, features and graphics must be assigned a symbol. There are a variety of symbol types you can create for displaying geoelements, with properties such as color, size, and style that you can modify. While all geoelements need a symbol in order to display, the symbol is applied differently for features and graphics.

- Features must be displayed using a renderer, which is basically a collection of symbols. The renderer is applied to the layer that contains the features. A renderer can also contain logic that determines which symbol to apply to each feature based on an attribute value.
- A graphic is more flexible. It can have a symbol applied directly, or can get its symbol from a renderer applied to the graphics overlay that contains it.

Symbols and renderers are described in the Symbols, renderers, and styles topic. For details about using a variety of available symbols and renderers to display geoelements, see Symbolize data.

How features and graphics relate to a map or scene

A map or scene contains layers, among other things. Feature layers in a map or scene are used to display features, feature layers are bound to a feature table. Features are not displayed using a graphics overlay.

A map view or scene view contains a map or scene, and graphic overlays, among other things. Graphic overlays in a map view or scene view always display graphics **on top of the map** and any layers the map contains. Graphics are not displayed using layers.

Where do features come from?

Features can be hosted in an online service, stored locally in a database, saved in a map or scene, or saved as a portal item. How you access features in your app affects how you make and manage edits to them.

Feature services

A feature service provides online access to features and can be hosted by ArcGIS Enterprise or ArcGIS Online. A feature service contains one or several collections of features. If you visit the REST Services Directory page for a feature service, you'll see the collections of features it contains listed under the **Layers:** heading, as shown in the following image. Features of different geometry types (point, line, and polygon, in other words) cannot exist in the same collection, so it's common to see features organized according to geometry type in addition to theme.

Copyright © 1995-2020 Esri. All rights reserved.

The feature service in the example contains several datasets, including three that describe **C2 Military Operations** using points, lines, and areas (polygons). The number in parenthesis indicates the index position inside the feature service. To access a specific set of features, append the index position to the URL for the service. **Hostile Units**, for example, can be accessed at `https://sampleserver6.arcgisonline.com/arcgis/rest/services/Military/FeatureServer/6`.

Feature tables

Features in your ArcGIS Runtime app are stored in feature tables of which there are many types. Features that come directly from an ArcGIS feature service are stored in a service feature table, which is created using the URL for the service and index position (see the URL above). Features read from a local geodatabase are stored in a geodatabase feature table; for more information on how to create a geodatabase see Take a layer offline. Static features stored in a map, scene, or portal item are stored in a feature collection table as part of a feature collection. Features from a web feature service are stored in a WFS feature table.

Since a feature service or local geodatabase can contain several sets of features (tables, in other words), you may need to create many feature tables in your ArcGIS Runtime app to represent all the datasets you need.

Types of features

Different feature table types return different feature objects. Feature tables that inherit directly from the FeatureTable base class return the Feature object type. Feature tables which inherit from ArcGISFeatureTable have additional capabilities such as attachments, these return ArcGISFeature objects.

For increased efficiency, the ArcGISFeature object implements the loadable pattern. When fetching features for rendering and query purposes, a minimum set of required fields are returned, such as identifiers, geometry, fields used for symbolizing, and so on. When you need all of the available fields you can simply load the feature.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
When querying features, you have the option to return your results as loaded features so all fields are immediately available.

Feature layers

It's not necessary to display the features you work with in your app. If you want your user to view (and interact with) features on a map or scene, however, add the feature table that contains them as a feature layer. A feature layer can display features from any type of feature table. See the Layers and tables topic for more information about creating and working with feature layers.

Features in a feature layer can use different request modes, including caching of features, for efficient display on the client. You should be aware of the various feature request modes, as the caching behavior of feature layers may affect the editing experience. Feature request modes are described in detail in the Layers and tables topic.

Caution:
A feature layer can also display data from a map service or a geodatabase in a mobile map package, in which case the data in the underlying feature table will be read-only.

Where do graphics come from?

Graphics are created by the developer and added to the application. They can be created from results from various operations such as querying, identifying, geoprocessing, geocoding or routing. They can also be created from external data sources, but if you want to persist the data in a map or scene then you must use features. Graphics can also be created by the developer as a result of a map click or touch.

When to use features

Persistence is a built-in characteristic of features. Features are persisted in a data store such as a database, a service, a map or a portal item. When this data store is available to multiple apps and users, a common set of data is available to all. Sharing data among users is a common use case for features.

Compare this to graphics, which reside in the memory of a single running app session. Graphics are created by the app during the session and are available only during that session.

You can publish features as part of a feature service. Layers in a feature service can be displayed in a map or scene, symbolized in various ways, and queried using attribute, spatial, or temporal criteria. The editing workflows and tools available in ArcGIS Runtime SDK also make it possible to expose editing functionality in your app. You can even add code to control the type of edits made and who gets to make them.

The following example adds a new feature layer to the map based on an online data source. The symbology (renderer) used to display the layer is defined with the feature service.

```
// define an online data source (ServiceFeatureTable)
var uri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/SF311/
FeatureServer/0");
var incidentsFeatureTable = new ServiceFeatureTable(uri);

// await loading of the table then check the load status
await incidentsFeatureTable.LoadAsync();
if(incidentsFeatureTable.LoadStatus == Esri.ArcGISRuntime.LoadStatus.Loaded)
{
    // create a new feature layer, pass the service feature table to the constructor
    var incidentsLayer = new Esri.ArcGISRuntime.Mapping.FeatureLayer(incidentsFeatureTable);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    // add the feature layer to the map
    MyMapView.Map.OperationalLayers.Add(incidentsLayer);
}
```

When to use graphics

Apps create graphics as needed, and graphics do not need to be persisted. Therefore, graphics are ideal for displaying things that are specific to the session, or anything that is displayed only temporarily. For example, the results of some tasks are returned as graphics, which your app may display on a graphics overlay.

The following are some common uses for graphics:

- Display text on top of a map or scene
- Highlight a section of the map or scene by overlaying a polygon graphic
- Display results from spatial analysis, such as buffer polygons created around features
- Display a route between two locations
- Display geometry drawn interactively by the app user
- Animate data items that change quickly, such as moving objects

The following example adds a provided polyline geometry as a graphic to a graphics overlay in the map view. The symbol for the graphic is created and applied explicitly.

```
private void DrawLineGraphic(Polyline polylineShape)
{
    // get the first graphics overlay in the map view
    var myGraphics = MyMapView.GraphicsOverlays[0];

    // define a line symbol (dashed blue)
    var lineSymbol = new SimpleLineSymbol();
    lineSymbol.Color = Color.Blue;
    lineSymbol.Style = SimpleLineSymbolStyle.Dash;
    lineSymbol.Width = 2;
          // create a new graphic; set the Geometry and Symbol
    var lineGraphic = new Esri.ArcGISRuntime.UI.Graphic();
    lineGraphic.Geometry = polylineShape;
    lineGraphic.Symbol = lineSymbol;

    // add the graphic to the graphics overlay
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    myGraphics.Graphics.Add(lineGraphic);
}
```

For more information about using graphics, see Add graphics and text to graphics overlays. For more information about using a graphics overlay, see Add graphics overlays to your app.

Related topics

Add features and graphics to a scene view
Add graphics overlays to your app
Add graphics and text to graphics overlays
Symbols, renderers, and styles

Copyright © 1995-2020 Esri. All rights reserved.

Symbols, renderers, and styles

This topic provides an overview of symbols, renderers, and styles and describes how they work in ArcGIS Runtime. The following topics describe other aspects of working with these objects:

- Symbol types—Describes the available types of symbols in ArcGIS Runtime
- Symbolize data—Describes how to symbolize your data, including code examples

A symbol defines display properties for features and graphics (collectively referred to as geoelements). A geoelement has a geometry (location and shape), optional descriptive information, and a symbol to define display characteristics such as color, size, border, transparency, and so on. It's important to remember, therefore, that a symbol is not the item being represented on the map. Instead, a symbol controls how those items (graphics or features) appear. The relationship is similar, for example, between the words you are reading now (the content) and the font that is used to display them (presentation). Changing the font style, size, and color will not change the meaning of the text but is likely to have an impact on the effectiveness of the presentation. Likewise, the quality of a map's presentation can improve with the proper use of symbols to convey information. Sometimes, the symbol used by a geoelement is contained in a renderer.

A renderer is a collection of one or more symbols. When applied to a layer or graphics overlay, a renderer displays geoelements using the appropriate symbol. If it contains more than a single symbol, a renderer uses logic to determine the symbol to apply to each geoelement, based on one or several attribute values. ArcGIS Runtime also contains a set of raster renderers for displaying rasters according to their cell values.

Some layer types don't support symbols or renderers, such as WMS and vector tile layers. These layers provide styles as an alternative method for changing how features are displayed. As with symbols and renderers, styles determine how a layer's content is presented.

Note:

You can apply a symbol directly to individual graphics. You can also use symbols to create a renderer for graphics overlays or feature layers. Symbols, however, cannot be assigned directly to a feature. Symbols can only be applied to features through the use of a renderer assigned to the layer that contains them.

Symbols

For features and graphics to appear on a map, they must be assigned a symbol. There are a variety of symbol types you can create to display geoelements, with properties such as color, size, and symbol style that you can modify. While each symbol type requires a specific geometry type (point, line, or polygon), you are not restricted to use those symbols exclusively for a given geometry. If you're symbolizing a line, for example, you can choose to use a marker symbol to display the line's vertices (points). Also, symbols may have different capabilities when used in 2D (map) or 3D (scene). To learn more about working with the available symbol types, see Symbol types.

Copyright © 1995-2020 Esri. All rights reserved.

ArcGIS Runtime uses the following two distinct models for symbols, based on the underlying data source:

- Simple symbols follow the web map specification. You can create these symbols through the simple symbology API, or get them from web maps and feature services when advanced symbology is turned off.
- Advanced symbols follow the ArcGIS Pro symbol model. You can create them through ArcGIS Runtime multilayer symbol classes, or get them from feature services, mobile style files, dictionary renderers, and mobile map packages.

Simple symbology is the symbology of the web map. When authoring maps in ArcGIS Pro as web maps, your symbols will be converted to simple symbols. In general, point symbols are converted to picture marker symbols optimized for the web, and line and polygon symbols are simplified while representing the original symbol as closely as possible.

If your app works primarily with web maps that you want to look the same throughout the platform, your app should use the simple symbols API. If you use multilayer symbols and try to save your map as a web map, the save will fail. Forcing it to save drops the symbols.

If your maps are used only with ArcGIS Runtime and ArcGIS Pro, you can use multilayer symbols. Multilayer (advanced) symbols are vectorized in these environments, thereby scaling better on devices with high resolution screens.

For details about specific symbols available in ArcGIS Runtime and examples of their use, see Symbol types.

Renderers

A renderer contains a set of symbols and controls how data in a layer (or graphics overlay) are displayed. Renderers use data values (from an attribute or raster cell) to determine the symbol to apply. There are a variety of renderer types, some for geoelements and some for rasters, each designed to use a different rendering logic.

Renderers are always used to symbolize feature or raster layers, since symbols cannot be applied directly to that data. A renderer can also be applied to a graphics overlay but may not be appropriate if the overlay has graphics of mixed geometry types. For such a scenario, applying the appropriate symbol directly to each graphic may be the preferred workflow.

Renderers can be updated at run time, allowing your user to dynamically visualize data in the map.

The following are the types of renderers available for geoelements in ArcGIS Runtime:

- Simple—Displays all features in a layer or graphics overlay using the same symbol. For example, display all points in the world cities layer as a small red square.
- Unique value—Applies a unique symbol for each specified value for an attribute (or combination of attributes). A unique value renderer can be based on any data type but is generally used with string attributes. For example, display points in the world cities layer using two symbols: a small gray triangle for features with a value of N for the `CAPITAL` attribute and a large yellow star for those with a value of Y.

Copyright © 1995-2020 Esri. All rights reserved.

- Class breaks—Symbolizes geoelements according to specific ranges of values for an attribute. The attribute used for a class breaks renderer must be numeric. For example, display world cities with three different symbols: a small blue circle for cities with a value between 0 and 100000 for the `POPULATION` attribute, a slightly larger blue circle for cities with a `POPULATION` value between 100001 and 2999999, and a larger blue circle for cities with a value over 3000000.
- Dictionary—Renders geoelements by constructing multilayer symbols from a style file and associated attribute values. This renderer is commonly used to display military symbology.

The following image shows a layer with a class breaks renderer. The renderer displays features as five classes of population, each with a different symbol (a darker or lighter shade of red).

The following raster renderers are available to control how raster data is presented.

- Hillshade—Creates a grayscale 3D representation of an elevation surface, with the sun's (hypothetical) position taken into account for shading the image. It can be applied to a raster layer created with single-band raster data.
- Blend—Blends a hillshade image (derived from the raster) with the original raster. This provides a look similar to the original raster but with some terrain shading, for a rich, textured look.
- Colormap—Provides a discrete mapping of raster pixel values to colors. All pixels matching the specified value are rendered using the mapped color. This can be useful for tasks such as land classification.
- Stretch—Displays continuous raster cell values across a gradual ramp of colors. Use the stretch renderer to draw a single band of continuous data. The stretch renderer works well when you have a large range of values to display, such as in imagery, aerial photographs, or elevation models.
- RGB—Uses the same methods as the stretch renderer but allows you to combine bands as red, green, and blue composites.

Copyright © 1995-2020 Esri. All rights reserved.

Styles

Some layers that don't support symbols and renderers, such as ArcGIS vector tiled layers and WMS, offer styles as a way to control the display of the features they contain. These layers use a default style and also provide the option to apply other available styles.

Note:

The styles described here are not the styles you can create using ArcGIS Pro (`.stylx` file). For information about reading symbols from such a style file, see Read symbols from a style file.

ArcGIS vector tiled layer styles

An ArcGIS vector tile layer consumes vector tiles and an associated style for drawing them. Because the style is separate from the underlying data, you can customize the style of an existing basemap layer. There are layers with many styles available through ArcGIS Online.

You can create your own style with the online style editor. Your customized vector layers can then be saved to and read from ArcGIS Online.

WMS styles

WMS servers provide clients with a list of supported styles for each layer. At run time, you can choose the style the WMS server uses to render map images. In general, styles are predefined and cannot be changed or added to.

The styles defined in the layer information can be inspected to determine the styles (if any) that are available. The style of a WMS sublayer can be set to one of the available styles.

Related topics

Add features and graphics to a scene view
Add graphics overlays to your app
Add graphics and text to graphics overlays
Features and graphics

Copyright © 1995-2020 Esri. All rights reserved.

Symbol types

This topic describes the available types of symbols in ArcGIS Runtime. The following topics describe other aspects of working with symbols:

- Symbols, renderers, and styles — Provides an overview of these objects and how they work to display different types of data
- Symbolize data — Describes how to symbolize your data, including code examples

A symbol defines display properties for features and graphics (collectively referred to as geoelements). A geoelement has a geometry (location and shape), and optional descriptive information. For features and graphics to appear on a map, they must be assigned a symbol. ArcGIS Runtime uses two different models for defining symbols in your map: Simple and advanced. In general, simple symbols are single-layer symbols that provide basic symbols, such as marker, line, fill, text, or picture. Advanced symbols are composed of one or several layers that can be defined individually and combined to create complex cartographic representations. Both of these are more fully described as follows:

- Simple symbols follow the web map specification and you can create and work with them through the simple symbol classes in the API. These are also the symbols you get from web maps or from feature services when advanced symbology is turned off. Simple symbols can be created for points (marker symbols), lines (line symbols), and polygons (fill symbols). Each of the simple symbol types provides an enumeration of pre-defined styles that can be applied to the symbol.
- Multilayer (advanced) symbols, accessed through multilayer symbol classes, follow the ArcGIS Pro symbol model. These symbols come from feature services (that use advanced symbology), mobile style files, the dictionary renderer, and mobile map packages. You can also build your own multilayer symbols for points, lines, and polygons in the map, as described in this topic.

Simple symbology is the symbology of the web map. When authoring maps in ArcGIS Pro as web maps, your multilayer symbols will be converted to simple symbols. In general, point symbols are converted to picture marker symbols optimized for the web, and line and polygon symbols are simplified while representing the original symbol as closely as possible. If you're authoring a feature service from ArcGIS Pro or ArcGIS Desktop, however, both the original symbols and the simplified symbols are stored. This allows clients that support advanced symbols to render the features as originally symbolized, while those that do not support advanced symbols (such as ArcGIS Online Map Viewer) can use the simple symbols for display. Having both sets of symbols allows you to retain the advanced symbology where available and still share the feature service as widely as possible.

If your app works primarily with web maps that you want to look the same throughout the platform, your app should use the simple symbols API. Otherwise, make sure your users understand that multilayer symbols render slightly differently on clients that don't support advanced symbology.

If your maps are used only with ArcGIS Runtime and ArcGIS Pro, you can use multilayer symbols exclusively. Multilayer (advanced) symbols are vectorized in ArcGIS Runtime clients, thereby scaling better on devices with high resolution screens.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
You can set `UseAdvancedSymbology` on the map's `LoadSettings` to control whether the map uses advanced symbols (when available) or always renders with simple symbols.

Multilayer (advanced) symbols

Multilayer symbols are based on a subset of the ArcGIS Pro symbology model. These symbols are composed of one or more symbol layers and can contain different symbol types within each layer with definable behaviors to get advanced cartographic effects. You can modify properties of individual symbol layers, or use properties of the symbol itself to apply a value (color or size, for example) to all the symbol layers it contains. With some exceptions and minor differences in behavior and appearance, these symbols can be used in both 2D (maps) and 3D (scenes).

Symbols used by web scenes are also multilayer but use a different specification than the ArcGIS Pro and ArcGIS Runtime symbols. When you read symbols from a web scene in your ArcGIS Runtime app, you get an ArcGIS Runtime representation of the multilayer symbol.

You can use ArcGIS Pro to author these complex symbol types and share them through feature services (if the service setting **Use Advanced Symbology** is **true**), mobile map packages, or mobile style files. The same multilayer symbols and the components used to build them are available in ArcGIS Runtime, allowing you to programmatically create and modify these symbols and apply them to geoelements (features or graphics) in your map.

Note:
Symbol layers in an ArcGIS Runtime multilayer symbol are in reverse order of how they appear in the JSON representation from the server.

The following summarizes features of multilayer symbology:

- Multilayer symbols are containers for (one or more) symbol layers.
- Each symbol layer uses a single geometry type (point, line, or polygon).
- When applied to a geoelement, invalid symbol layers for a particular geometry type are ignored.
- Symbol layers behave differently depending on the type of geometry they are rendering. When symbolizing a polygon feature, for example, a `StrokeSymbolLayer` defines the symbol for the polygon outline.
- Each multilayer symbol class includes helper functions to change common properties (such as color and size) without having to iterate through all the layers the symbol contains.
- Multilayer symbols share the same base class as the simple symbols (`Symbol`) and can be used interchangeably in your ArcGIS Runtime SDK apps.

Symbol layers

Symbol layers are the fundamental component used to build multilayer symbols. Similar to how a collection of map layers are overlaid to display a map, symbol layers in a multilayer symbol are overlaid to display a symbol. The available types of symbol layers correspond to the geometry type they are designed to render: marker layers for point, stroke layers for line, and fill layers for area. Each layer type has a unique set of properties to control its display.

A symbol can contain one or several symbol layers of the same or different type. A symbol used to render polygons, for example, might contain a fill symbol layer to render the polygon interior, a stroke symbol layer for rendering the polygon outline, and a marker symbol layer to render polygon vertices. A symbol to render a polyline might contain several stroke layers

Copyright © 1995-2020 Esri. All rights reserved.

of varying widths. If a symbol contains a symbol layer that is invalid for the type of geometry it's rendering (a fill layer on a polyline, for example), that symbol layer is ignored.

The following table lists symbol types and symbol layers:

Symbol type	Typical symbol layers	Role
Point symbol	Marker layers	Draw relative to the point geometry
Polyline symbol	Marker layers	Draw at specified locations relative to the line geometry
	Stroke layers	Draw relative to the line geometry
Polygon symbol	Marker layers	Draw relative to the polygon outline or in the interior, depending on placement setting
	Stroke layers	Draw relative to the polygon outline
	Fill layers	Draw relative to the polygon interior

The following examples from ArcGIS Pro illustrate the use of marker, stroke, and fill layers to build symbols. A marker symbol is used in each example, participating in the symbols for points, lines, and polygons.

Point symbol	Polyline symbol	Polygon symbol

Marker symbol layer

Marker symbol layers draw a shape or picture at a specific location relative to point geometry. They are most often used in point symbols, but they can also be included in polyline and polygon symbols to draw a symbol at locations along lines and outlines or within areas (the center of a polygon, for example).

ArcGIS Runtime includes the following types of marker symbol layers:

- Vector marker symbol layer—Vector geometry defines the shape and appearance of the marker.
- Picture marker symbol layer—Uses an image file (specified using a URI or image data) to define the marker.
- Model symbol layer (scenes only)—Uses a true three-dimensional graphical model to define the marker. The model used is derived from a URI to a model in a supported Open Asset Import Library (Assimp) format.

Should you use a character marker symbol or a vector marker symbol? While a simple text symbol (`TextSymbol`) can be used to display a character as a marker, proper display is dependent on the referenced font being available on the machine or device running the app. If the exact font is not found, which is likely to occur on many mobile devices, a fallback mechanism is used to find the best available font to use. A character marker symbol authored from ArcGIS Pro, however, is represented by a vector

Copyright © 1995-2020 Esri. All rights reserved.

marker symbol. This symbol stores the geometry that represents the character, is not dependent on the font to render the shape of the marker, and displays as expected on all devices.

Line symbol layer

Line symbol layers define the appearance of a (2D or 3D) line. They might include effects to provide a dash pattern or define properties for the line end cap style. For display in a scene, they can define the 3D rendering style, such as tube, strip, or wall appearance. At version 100.5.0, `StrokeSymbolLayer` is the only type of line symbol layer.

Stroke symbol layers draw lines and outlines using a solid or dashed symbol. They are most often used in line symbols or in polygon symbols as outlines.

Fill symbol layer

Fill symbol layers are components of symbols that cover areal geometries. They are most often used in polygon symbols.

ArcGIS Runtime includes the following types of fill symbol layers:

- Solid fill symbol layer—Fills polygonal geometry with a single solid color (that may have a level of transparency applied).
- Picture fill symbol layer—Uses an image file (specified using a URI or image) to define the fill.
- Hatch fill symbol layer—Defines a pattern of uniform parallel lines.
- Material symbol layer—Material fills are only available in mesh symbols that are applied to 3D objects. The color applied to the object is based on the specified material mode.

Symbol and layer property interaction

Most `SymbolLayer` properties are only accessible on the symbol layer. There are some properties, however, that are also exposed for the symbol as a whole. Properties such as `Color` and `Size`, for example, can be modified either for an individual symbol layer, or for the symbol itself. For such properties, the value provided for the symbol may have an effect on the corresponding property for the symbol layers it contains. Similarly, the property value for a multilayer symbol might be determined from the values of the individual symbol layers it contains.

Some common symbol properties and their behavior in a multilayer symbol are described in the following sections. Examples are provided for each property to help illustrate how these properties interact between individual symbol layers and the symbol that contains them.

Note:
There are no notifications from a symbol or symbol layer when a property has been changed. If your app needs to respond to property changes in a symbol—as a result of a propagated change, for example— you will need to reread property values to identify changes.

Color

Most symbol layers have some type of color property that can be set directly or through the same property on the symbol that contains them. Marker, stroke, and fill symbol layers have a color property, and picture symbol layers have a tint color. Tint color is applied as a tint to the entire image in a picture symbol layer.

When getting the color (or tint color) from a symbol, the consensus value is reported. If there is no consensus (one or more layers have different colors), a null value is reported for the color. Likewise, setting a value for color on a symbol will apply that color to all layers the symbol contains.

Copyright © 1995-2020 Esri. All rights reserved.

If you don't want changes to the symbol color to propagate to some of the symbol layers it contains, you can lock the color (or tint color) for those layers. If a symbol layer is color-locked, changes to the parent symbol will not affect the symbol layer. This also means that the layer will not be considered when determining a consensus color for the symbol as a whole.

The following example illustrates the behavior of the color property for a multilayer polygon symbol:

1. A multilayer polygon symbol has two symbol layers: layer A is a picture marker symbol with a blue tint color, and layer B is a solid fill symbol layer with a blue color. Neither layer is color-locked.
2. The polygon symbol reports a color of blue, since both layers are blue.
3. A tint color of red is set for the picture marker symbol layer. Since there is no longer a consensus for layer colors, the polygon symbol reports null for the color.
4. The picture marker symbol layer is color-locked. The polygon symbol now reports a color of blue, since that is the consensus color for all color-unlocked layers.
5. The polygon symbol color is set to green. The fill symbol layer color updates to green, and (because it's locked) the picture marker symbol layer remains red. The polygon symbol now reports a color of green.

Setting a color for a symbol will propagate that color to all color-unlocked symbol layers it contains, regardless of the number of nested levels of symbols and layers there are. Similarly, all nested symbols and layers are considered when determining a consensus color for a symbol.

Size

When getting the size from a multilayer point symbol, the value reported is the largest size of all the symbol layers it contains. The symbol size is updated when its symbol layer sizes change (if there is a new largest layer size). When setting a new size at the point symbol level, the size of the largest symbol layer is given the provided value, and all the smaller layers are sized proportionately. An individual symbol layer can be given a specific size without affecting the other layers.

The following example illustrates the behavior of the size property for a multilayer point symbol:

1. A multilayer point symbol has two marker symbol layers: layer A and layer B. Both have a size of 10.
2. The point symbol reports a size of 10 (the largest size for all layers).
3. The size property on the symbol is set to 15. Both layers in the symbol now have a size of 15.
4. The size of marker symbol layer A is set to 10. The size of marker symbol layer B is unchanged (15). The size for the symbol also remains 15, since that is the largest size for the layers.
5. The size of the symbol is set to 20. The largest marker symbol layer (layer A) updates from 15 to 20. Marker symbol layer B is updated from the current value of 10 using the same proportion: 10 * (1 + 1/3) = 13.333.

Copyright © 1995-2020 Esri. All rights reserved.

This behavior does not apply to nested symbols. You can have a point symbol, for example, that has a vector marker symbol layer that in turn contains a point symbol with its own set of marker symbol layers. The size of the nested point symbol (and any additional layers it may contain) is not affected by changes to the size at the top level point symbol. Likewise, changes to the size of the nested point symbol are not communicated to the top level symbol for determining the top level symbol size.

Angle

Angle for multilayer point symbols and heading for marker symbol layers is measured in degrees counterclockwise from the east, from 0 to 360. The angle for a multilayer point symbol defaults to 0. Marker symbol layers can have a separate heading value defined (which is also 0 by default). Setting a heading for a marker symbol layer does not affect the angle of the point symbol. When an angle value is set for the point symbol, however, that value is applied cumulatively to all the marker symbol layers it contains. The heading for individual symbol layers must be re-fetched in order to see the updated value.

The following example illustrates the behavior of the angle property for a multilayer point symbol:

1. A multilayer point symbol contains a single marker symbol layer. The symbol and marker symbol layer both have the default value of 0 for angle and heading respectively.
2. The marker symbol layer heading is set to 90. The angle value reported by the point symbol is unchanged (0).
3. An angle of -45 is applied to the point symbol. The marker symbol layer heading is updated with the delta of the current marker symbol layer heading and the point symbol angle: 90 -45 = 45.
4. When the symbol is rendered, the marker symbol layer will be rotated 45 degrees counterclockwise.

Width

The behavior of width for a polyline symbol (and the stroke symbol layers it contains) is identical, in most instances, to the behavior described previously for size with a point symbol. When getting the width from a multilayer polyline symbol, the value reported is the largest width of all the symbol layers it contains. The symbol width is updated when its symbol layer widths change (if there is a new largest layer width). When setting a new width at the polyline symbol level, the width of the largest symbol layer is given the provided value, and all the smaller layers are sized proportionately. An individual symbol layer can be given a specific width without affecting the other layers.

Width and size properties can have crossover effects for some symbols that contain both marker and stroke layers as follows:

- Polyline symbol with vector marker symbol layers and solid stroke symbol layers
  - Symbol width is equal to the maximum value for marker size and stroke width.
  - Setting the symbol width adjusts the width of stroke symbol layers as well as the size of marker symbol layers.
- Point symbol with vector marker symbol layers and solid stroke symbol layers
  - Setting the symbol size adjusts the size of marker symbol layers and the width of stroke symbol layers.
  - Symbol size is equal only to the maximum size of all marker symbol layers; stroke symbol layer width is not considered.

Copyright © 1995-2020 Esri. All rights reserved.

Unknown symbol layers or properties

The multilayer symbol specification is complex and continues to evolve outside of ArcGIS Runtime. The potential exists that some symbol layers (or properties) will not expose an API or will not be known by ArcGIS Runtime. Depending on the problem encountered, ArcGIS Runtime will handle this situation in one of the following ways:

- If ArcGIS Runtime reads a symbol that it can render, but for which no public API is exposed, the base class for the appropriate symbol layer is returned in the layer collection.
- The multilayer symbology specification for an existing symbol layer type might be updated to include properties that are not recognized by ArcGIS Runtime. When reading such a symbol layer, ArcGIS Runtime attempts to create it with the properties it understands. The symbol layer renders, but properties that cannot be identified are ignored.
- In the event that a new (unrecognized) symbol layer is encountered by ArcGIS Runtime, the symbol is ignored and is not added to the symbol layer collection.

Simple (web) symbols

Depending on the type of geometry you want to symbolize, there are options available for defining your symbol. Points, lines, and polygons all have a basic symbol you can use: `SimpleMarkerSymbol` , `SimpleLineSymbol` , and `SimpleFillSymbol`, respectively. You can also symbolize points using an `PictureMarkerSymbol` and polygons with `PictureFillSymbol`.

The following table describes the simple symbols available in the API with the compatible geometry types and a brief description:

Symbol/Class	Geometry	Description
`SimpleMarkerSymbol`	point, multipoint	Symbolizes points or multipoints with basic shapes
`PictureMarkerSymbol`	point, multipoint	Symbolizes points or multipoints with images
`SimpleLineSymbol`	polyline	Symbolizes polylines with predefined styles
`SimpleFillSymbol`	polygon, envelope	Fills polygons or envelopes with a color and predefined styles
`PictureFillSymbol`	polygon, envelope	Symbolizes polygons with tiled images
`TextSymbol`	point, multipoint, polyline, polygon	Displays text

All marker symbols provide the ability to set an angle for rotating the symbol. This may be useful for showing things such as direction (for wind measurements or moving vehicles, for example). In addition, marker symbols have an angle alignment property that specifies whether your marker symbol remains screen-aligned or map-aligned when the map rotates. The default for all marker symbols is screen-aligned, meaning that the symbol will not rotate with the map. When you set the angle alignment property to map-aligned, the symbol will rotate with the map. This is important in cases where the symbol's angle indicates a direction on the map. You may also have the ability to define an x or y offset, meaning the symbol displays at a specified distance from the location of the feature it represents. This can be useful when working with text.

The simple symbol types (`SimpleMarkerSymbol` , `SimpleLineSymbol` , and `SimpleFillSymbol`) allow you to quickly create a symbol by choosing a style (for example, solid or cross hatched fill for polygons, solid or dashed line for polylines, and square or circle marker for points), setting a color, and defining a size (for markers) or width (for lines). For markers and fills, you can also define an outline symbol (using a line symbol).

Copyright © 1995-2020 Esri. All rights reserved.

Picture symbols

Picture symbols, such as `PictureMarkerSymbol` and `PictureFillSymbol`, allow you to use an image to symbolize a feature. The image can be stored locally, or originate from an online source. The size of the image can be specified in your platform's device independent units (DIPs) in the same way as your other application resources.

Picture marker symbols can be created from an image on disk, such as a JPEG file, or from the URL to an image on a remote machine accessible without security. Remote images are downloaded once per application and cached for future use; for this reason they are loadable resources.

You can create picture marker symbols in the following ways:

- From an image on the file system
- Use a URL to an image
- With a raw JSON string that defines an image
- From an asset (resource) in your project

See Symbolize data for examples of creating picture marker symbols.

3D-specific symbols

Graphics in a scene view graphics overlay can use the same (2D) symbols discussed in this topic. ArcGIS Runtime also provides 3D-specific symbols for graphics in a scene view.

- Simple marker scene symbols—Cone, Cube, Cylinder, Diamond, Sphere, and Tetrahedron
- Stroke symbol layer 3D line styles—Strip, Tube
- Model 3D symbols—Displays a 3D model file

Copyright © 1995-2020 Esri. All rights reserved.

Shape-based 3D symbols

Model 3D symbols

Simple marker scene symbols provide a variety of styles (shapes) that can be applied. You can also set the size, color, and anchor position used for the symbol. To further control 3D display, you can also set heading, pitch, and roll.

Lines can be rendered in a scene view using a stroke symbol layer with either a `Strip` or `Tube` 3D line style. A strip displays a line that conforms to the surface of the scene, and a tube draws as a cylinder.

Note:
The dimensions of simple marker scene symbols are defined in meters. For 3D lines, strip style lines are defined using device-independent pixels (DIPs) and tubes are defined in meters.

A model symbol provides a realistic three-dimensional visualization to symbolize scene features. These models define the geometry and surface textures of real-world items such as aircraft. Create a `ModelSceneSymbol` using a 3D model file, passing the path to the file and a scale factor to the model symbol constructor. Depending on the model file's default orientation, you may have to rotate the symbol to get your model in the desired orientation.

The supported model file types for 3D model symbols are:

- 3D Max (`.3ds`)
- Collada (`.dae`)
- Filmbox (`.fbx`)
- GL Transmission Format (`.gltf`, `.glb`)
- Wavefront (`.obj`)

Note:
Model scene symbols cannot be used with a static graphics rendering mode.

Copyright © 1995-2020 Esri. All rights reserved.

See Create 3D symbols for examples of creating graphics with 3D symbols.

Distance composite symbols

A distance composite symbol is a collection of several symbols. It determines which symbol to display based on the current camera distance to the graphic. While model symbols can accurately represent real-world objects when viewed close up, they may not be visible when viewed at a distance (unless the size of the object is exaggerated). A distance composite symbol allows you to display a different symbol based on the camera distance to the graphic. They are commonly used to show more simple symbols at a great distance and more complex symbols when the camera approaches the graphic, as illustrated by the following images.

When the camera is far away from the point, a red simple marker symbol displays.

As you zoom closer to the point, the symbol renders as a cone pointing in the aircraft's direction of travel.

When viewed even closer, the point displays as a model symbol, which is appropriate at this distance from the camera.

Copyright © 1995-2020 Esri. All rights reserved.

See Create 3D symbols for examples of creating graphics with 3D symbols, including distance composite symbols.

Copyright © 1995-2020 Esri. All rights reserved.

Tasks and jobs

Tasks are bound to online or offline data or services, and provide methods to perform asynchronous operations using those resources. By using tasks you can:

- Download, collect, and update geographic information using `GeodatabaseSyncTask`
- Download and display tiled basemaps using `ExportTileCacheTask`
- Locate addresses and find addresses from map locations using `LocatorTask`
- Calculate point-to-point or multi-stop routes and get driving directions using `RouteTask`
- Perform complex GIS analysis by executing geoprocessing models using `GeoprocessingTask`

Tasks either return their results directly from asynchronous methods on the task, or make use of jobs to provide status updates and results.

Tasks

Some operations return their results directly from asynchronous methods on the task, for example `LocatorTask.GeocodeAsync` and `RouteTask.SolveRouteAsync`. For more complex or longer running operations, tasks make use of jobs instead.

To use tasks that return results directly:

1. Create the task by initializing it to use the required data or service.
   - Some operations can work both online and offline.
2. Define the task inputs.
   - Some operations require only simple value inputs (for example a simple geocode operation may only require an address string as input).
   - Others require parameters to be defined (for example, to limit a geocode operation to a specific country).
3. Call the async operation method, passing in the inputs you defined.
4. Use the results from the operation as required, for example to display geocode results on a map.

The following code creates a `LocatorTask` to geocode an address entered by the user. The best match is used to create a graphic to display on the map.

```
// Create a new LocatorTask from a geocode service endpoint
var locatorUri = new Uri("http://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer");
LocatorTask onlineLocator = new LocatorTask(locatorUri);
  // If the address text box is not empty, run the geocode task
if (!string.IsNullOrEmpty(FromAddressTextBox.Text.Trim()))
{
    // Pass the address to the GeocodeAsync method and get the results
    IReadOnlyList<GeocodeResult> matches = await
onlineLocator.GeocodeAsync(FromAddressTextBox.Text);

    // Get the first match (one with the highest score)
    GeocodeResult bestMatch = matches[0];
          // Use geometry and attributes from the match (and a symbol created earlier) to create
a new graphic
    Graphic fromGraphic = new Graphic(bestMatch.DisplayLocation, bestMatch.Attributes,
fromPointSymbol);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    // Add the graphic to an existing graphics overlay in the map view
    MyMapView.GraphicsOverlays[0].Graphics.Add(fromGraphic);
}
```

Some Tasks are loadable, and will load themselves when you call an asynchronous method (like that shown above) that requires the task to be in a loaded state. Loadable objects are generally required to be loaded before their properties are filled out.

Define input parameters

Tasks offer numerous options that allow you to tailor the operation to your requirements. For example, when geocoding you can restrict your search to a specific area, country, category of place, and/or number of results. When an author publishes a service or packages a resource, they can choose default values for these options that suit the specific data or the most common use case for the service.

To find these default parameter values, tasks provide async methods that create parameters objects, initialized with these service-specific values. You can then make any changes to the parameter values, before using to execute an operation. Creating parameter objects in this way can be especially useful for operations with many options, such as solving a route.

The following example gets the default parameters for a route task, then explicitly sets values for some properties.

```
// get the default route parameters
var routeParams = await routeTask.CreateDefaultParametersAsync();

// explicitly set values for some parameters
routeParams.ReturnDirections = true;
routeParams.ReturnRoutes = true;
routeParams.OutputSpatialReference = MyMapView.SpatialReference;

// solve the route with these parameters
RouteResult routeTaskResult = await routeTask.SolveRouteAsync(routeParams);

// ... work with results ...
```

Alternatively, some parameters objects have constructors that you can use if you know the values of all the input parameters you want to use. This can be more efficient where parameter settings are simple.

For example, the following code creates a new `GeocodeParameters` that restricts the search country to **France** and limits the results (matches) to a maximum of **5**.

```
GeocodeParameters geocodeParams = new GeocodeParameters();
geocodeParams.CountryCode = "France";
geocodeParams.MaxResults = 5;
```

Work online or offline

Many tasks can work either online by using services, or offline by using local data and resources. For example, you can geocode an address by using the default Esri geocoding service, your own geocoding service, a locator file (`.loc`) or a mobile map package (`.mmpk`).

```
// Create a new LocatorTask from a geocode service endpoint
var locatorUri = new Uri("http://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer");
LocatorTask onlineLocator = new LocatorTask(locatorUri);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create an offline locator from a local .loc file - coverage will depend on the packaged
locator dataset...
LocatorTask offlineLocator = new LocatorTask(localLocatorUri);

// Mobile map packages can also contain locators - use this to get a reference to an offline
locator...
MobileMapPackage mmpk = await MobileMapPackage.OpenAsync(localMmpkPath);
LocatorTask mmpkLocator = mmpk.LocatorTask;
```

Tasks and jobs

Some tasks expose operations that have multiple stages (like preparing and downloading a geodatabase), and can generate multiple progress messages (such as percentage complete). These types of tasks are always bound to ArcGIS Server or Local Server. An example is `GenerateGeodatabase` on `GeodatabaseSyncTask`.

Instead of returning results directly, these tasks make use of jobs to monitor status, return progress updates, and return their results. Each `Job` represents a specific operation of a task. Jobs are useful for longer-running operations, because they can also be paused, resumed, and cancelled. Your app can support a user action or host OS terminating a running job object, and then re-create and resume the job later.

To use operations like these:

1. Create the task by initializing it to use the required data or service.
2. Define the input parameters for the task.
3. Call the async operation method to get a job, passing in the input parameters you defined.
4. Start the job.
5. Optionally, listen for changes to the job status and check the job messages, for example to update a UI and report progress to the user.
6. Listen for the job completion and get the results from the operation. Check for errors in the job, and if successful, use the results.

The following example creates a task to synchronize local edits to a service. The `JobChanged` event is handled to report success or failure of the job.

```
// Function to submit a geodatabase synchronization job
// URL for the feature service and the path to the local geodatabase are passed in
public async Task SyncronizeEditsAsync(string serviceUrl, string geodatabasePath)
{
    // create a sync task with the URL of the feature service to sync
    var syncTask = await GeodatabaseSyncTask.CreateAsync(new Uri(serviceUrl));

    // create sync parameters
    var taskParameters = new SyncGeodatabaseParameters()
    {
        RollbackOnFailure = true,
        GeodatabaseSyncDirection = SyncDirection.Bidirectional
    };
    // open the local geodatabase
    var gdb = await Esri.ArcGISRuntime.Data.Geodatabase.OpenAsync(geodatabasePath);

    // create a synchronize geodatabase job, pass in the parameters and the geodatabase
    SyncGeodatabaseJob job = syncTask.SyncGeodatabase(taskParameters, gdb);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    // handle the JobChanged event for the job
    job.JobChanged += (s, e) =>
    {
        // report changes in the job status
        if (job.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Succeeded)
        {
            // report success ...
            statusMessage = "Synchronization is complete!";
        }
        else if (job.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Failed)
        {
            // report failure ...
            statusMessage = job.Error.Message;
        }
        else
        {
            statusMessage = "Sync in progress ...";
        }
    };

    // await the completion of the job
    var result = await job.GetResultAsync();
}
```

Calling `Job.Status` retrieves the current point in the job's workflow. Started jobs periodically receive job changed events; this happens with decreasingly frequency as a job progresses; more than one `JobMessage` may have been added to the job for each call. The job done listener is called as soon as the job is complete, for both successes and failures. Completed jobs, whether they have succeeded or failed, cannot be restarted.

Report job progress

A job represents an asynchronously running operation that might take some time to finish. As described previously, you can monitor changes to job status for notification when a job has completed, failed, or been cancelled, but what about the time in-between? Users may become frustrated waiting for a long job to complete without getting feedback on it's progress. Fortunately, jobs provide a mechanism for reporting the current progress (percentage complete) for the running operation they represent.

As the job runs it raises a `ProgressChanged` event. You can get the current progress of the job at any point from the job's `Progress` property, an integer representing the percentage of the operation that has been completed. This allows your app to provide more specific information about the status of a running job using UI elements like progress bars, for example.

The following example updates a label in the UI to show the percentage complete for the job.

```
// Execute a task to return the Job
GenerateOfflineMapJob job = task.GenerateOfflineMap(parameters, packagePath);

// Hook into the ProgressChanged event to track the current progress
job.ProgressChanged += (s, e) =>
{
    // Note: The event might be raised in a background thread.
    //       Make sure updates to UI elements are pushed to the UI thread.
    Dispatcher.Invoke(() =>
    {
        // Update a UI element to show the updated progress value (percent complete)
        PercentageLabel.Text = "Operation is %" + job.Progress.ToString() + " complete ...";
    });
};
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Run the job and await the results asynchronously
GenerateOfflineMapResult results = await job.GetResultAsync();
```

Pause, resume, or cancel job

Jobs are designed to effectively handle a user exiting an app while the job is running, and to handle the app being terminated by the host operating system. Jobs also deal with explicit pausing and cancellation of the operation.

Cancel a job

Sometimes the results of a job are no longer required. For example, a user could change their mind about the area of a tile cache they want to download and want to cancel the job and start over.

Calling `Job.Cancel` on a job immediately changes its status to `JobStatus.Failed` and adds additional information to the job error object. The job completion listener will be called. The error object indicates the error domain and code, which allows you to identify when cancellation has occurred.

The following example shows code you would add to the `JobChanged` handler to check for a `System.OperationCanceledException` when the job fails. Canceling the job might require additional cleanup of incomplete data or other code on the client.

```
if (job.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Failed)
{
    // get the exception from the job
    Exception jobException = job.Error;

    // check for cancellation
    if (jobException is OperationCanceledException)
    {
        statusMessage = "Job was canceled.";
    }
    else
    {
        // report the exception message
        statusMessage = "Job failed: " + jobException.Message;
    }

    // ... code here to clean up (delete incomplete files, for example) ...
}
```

Cancelled jobs cannot be restarted, although you can start a new job by re-using the same parameters. Cancelling a job does not necessarily stop any server-side processes, because not all services support cancellation on the server side.

Pause and resume a job

Jobs can be long running operations, so there is no guarantee that they will be completed while the app is running. You can pause a job explicitly using `Job.Pause`. For example when an app is backgrounded and does not have permissions for background operation. Pausing may also be useful if a user wishes to temporarily stop network access for any reason.

Job changed messages will not be received for a paused job. Pausing a job does not stop any server-side processes from executing. While a job is paused, outstanding requests can complete, and therefore it's possible that a resuming a job will result in it having a different state to when it was paused.

Copyright © 1995-2020 Esri. All rights reserved.

You can serialize a job to JSON to persist it if your app is backgrounded, or the process is otherwise terminated. When you deserialize it again, the job will be in the `JobStatus.Paused` state regardless of its state when serialized, and should be restarted to resume listening for completion. The job changed listener can be a good place to update the job JSON for storage by your app.

The following example shows code to save the state of a job in its `JobChanged` event handler. Additional code could be used to save this text (JSON) as a local file for deserialization later.

```
// handle the JobChanged event for the job
job.JobChanged += (s, e) =>
{
    // when the job changes, save its current state to JSON
    _jobJson = job.ToJson();

    // ... code to check job status ...
};
```

The Job can be deserialized from a JSON string and resumed if necessary. You must re-wire event handlers to monitor the job status (`JobChanged`, for example), as shown in the following example.

```
private void DeserializeJob(string jobJson)
{
    // recreate the SyncGeodatabaseJob from the JSON representation
    SyncGeodatabaseJob job = SyncGeodatabaseJob.FromJson(jobJson);
          // handle the JobChanged event for the job
    job.JobChanged += (s, e) =>
    {
        // ... code to check job status, and so on ...
    };

    // start (resume) the job
    job.Start();
}
```

Loss of network connection

Additionally, jobs using services are designed to handle situations where network connectivity is temporarily lost without needing to be immediately paused. A started job will ignore errors such as network errors for a period of up to 10 minutes. If errors continue for longer, the job will fail and the message will indicate the loss of network connection.

To deal with partially-connected workflows, you can serialize and pause a job when your app detects that network connectivity has been lost for a few minutes to avoid job failure purely due to this lack of network connectivity (as failed jobs cannot be restarted). The job can then be deserialized and resumed when connectivity is reinstated.

Copyright © 1995-2020 Esri. All rights reserved.

Offline

Offline maps and scenes allow your users to continue being productive even when their network connectivity is poor or non-existent. Your apps can explore maps, collect information, edit asset data, find places and route to new locations, all while disconnected from the Internet. You can choose to synchronize your data edits with other users when a connection is re-established.

ArcGIS Runtime SDK for .NET supports a range of workflows to help you take your maps and scenes offline. These workflows fall into two broad categories

- the Services pattern supports a dynamic workflow where multiple users
  - are occasionally online so that offline maps and data can be downloaded to the device
  - receive regular data updates provided by online services
  - can edit and share their updates with other users
- the Desktop pattern supports users who
  - are never online or access is very limited
  - work with read only data that rarely changes

Services pattern

The services pattern allows you to build apps that can download specific maps, layers and data, on request, to the device. Users can edit their operational data offline and synchronize it with other users when connectivity is restored. Choose a workflow that best matches your business needs:

- Preplanned workflow: A manager authors and generates a map area, ahead of time, so that any field worker can download it to their device and work with it offline.
- On-demand workflow: The field worker defines a map's area of interest, generates the map content and downloads it to their device.
- Take individual layers offline: Field workers take individual layers offline and use these to construct the base map, operational layers and maps within their app.

Note:
For these Services patterns, you must ensure that all relevant layers can be taken offline and ensure that any feature layer's sync capability is enabled. Learn more in Manage hosted feature layers - ArcGIS Online Help.

Preplanned offline workflow

The preplanned workflow allows a manager to define a map area and generate the map content so that it can be downloaded by many field workers. For example, a gas utility company may have a maintenance crew that inspects pipeline gauges throughout a city. The manager of the crew can prepare a set of map areas prior to the week's work so that individual team members just download the map areas to their devices at the beginning of the week. Team members will be able to work offline to locate the inspection gauges. There are two phases:

The work or crew manager produces each preplanned map area as follows:

1. Create a map area by specifying an area of interest for each work/inspection zone .
2. Generate the map area content so that it is ready for download.

The crew downloads the work/inspection map areas to their device using your app. The app code must:

Copyright © 1995-2020 Esri. All rights reserved.

1. Query the online map to find the map areas.
2. Select a map area.
3. Download the map area, along with the map content, onto the user's device.
4. Display the offline map and allow the user to edit the data, if required.
5. Go offline with the map

For step details see Take a map offline - preplanned.

On-demand offline workflow

The on-demand offline workflow allows your user to define the exact area of a map they wish to take offline. For example, if a gas leak is identified at an address an emergency crew member can generate a map showing the address and surrounding road network along with the gas network, pipes, gauges, and so on. They download the map and head out into the field to resolve the problem and collect relevant information.

To download this offline map to the user's device your app needs to:

1. Present a UI to allow the user to define an area of interest.
2. Generate the map content for that area of interest.
3. Download the map content to the user's device.
4. Display the offline map and allow the user to edit the data, if required.
5. Go offline with the map.

For step details see Take a map offline - on-demand.

Take individual layers offline

You can also build an app that takes individual layers offline using this API. The app can then control how the map is built, which layers to combine and how to render the information. If you require this level of control then see Take a layer offline.

Desktop pattern

ArcGIS Pro allows you to create mobile map and scene packages that can be taken offline. An advantage of this approach is that you can prepare the data once, using ArcGIS Pro, shared it with your ArcGIS organizational account and download or just copy it directly to a device. The desktop pattern is appropriate where the data is fairly static and does not change on a regular basis. This content can include features, tabular data, tile caches, network datasets for directions and locators that can be packaged into a single mobile map or scene package file.

Create an offline map or scene

Use ArcGIS Pro to create mobile map package (`.mmpk`) or mobile scene package (`.mspk`) files. Each package can contain multiple maps or scenes, their associated layers and data, and optionally networks and locators. After the mobile package is downloaded or copied onto your device you can open it using the `MobileMapPackage` or `MobileScenePackage` class. For more details see see take a map offline - ArcGIS Pro and take a scene offline - ArcGIS Pro.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
With ArcGIS Pro 2.6 (or later), you can include online layers and tables in your map and scenes, such as traffic information, weather conditions, or a large company data-set. If your user's device has network connectivity they can take advantage of these online layers and tables. If the connection is unavailable, users can continue to work with the maps and scenes that are stored locally in the package. Online layers and tables may require that you handle authentication in your app.

For more information about creating packages in ArcGIS Pro see Create a mobile map package or Create a mobile scene package. If you'd like a ready-to-use and regularly updated network dataset (and locator) for your area of interest, you can license StreetMap Premium data (in mobile map package format). For details, see Add StreetMap Premium data.

Create offline layers

You can also create individual layer packages directly from ArcGIS for Desktop or ArcGIS Pro. This approach will provide you with more control over when individual packages are generated and you can work with each package separately.

ArcGIS Desktop can also be used for creating runtime content but with limited capabilities. For more information see Creating ArcGIS Runtime content in the ArcMap help.

See Work with offline layers for more information on working with data such as geodatabases, tile packages, geopackages, and raster datasets.

Enabling data and licensing

Operational data

To allow people to take a hosted feature layer offline and work with it while disconnected from the network you must enable synchronization on the hosted feature layer. Through the synchronization process, you can control whether a user can upload their edits, download others' edits, or both. Synchronization allows the app to maintain up-to-date data in the offline map. When more than one user is editing the same feature and the edits conflict, the last edit synchronized to the service overrides the others.

For details on using hosted feature layers, see Manage hosted feature layers in the ArcGIS Online help. For details on using ArcGIS Server feature services, see Prepare data for offline use and Tutorial: Configure feature service data for offline use in the ArcGIS Server documentation.

Note:
If you need to provide offline routing or geocoding in the app, create the locator and network datasets using the Desktop pattern.

Basemap data

For basemap data, ArcGIS tiled map services, ArcGIS image services, and ArcGIS Online basemaps can be downloaded to a user's device in the form of tile packages (`.tpk` or `.vtpk` files). This can be from your own ArcGIS Server or a hosted tile layer. For details on enabling export of tiled map services, see Allow clients to export cache tiles in the ArcGIS Server documentation.

ESRI provides a set of tiled basemaps and a set of vector tiled basemaps that can both be exported as tiled packages for offline use. These services require an ArcGIS Online organizational subscription or an ArcGIS Developer account and do not consume credits. The licensing details for using these basemaps offline are described in Use tile packages from ArcGIS Online basemaps.

Copyright © 1995-2020 Esri. All rights reserved.

Licensing

Apps that use offline mapping require the following license levels:

To view offline basemaps, view and download updates to offline feature data contained in a geodatabase, and work with offline routing and geocoding the app can be licensed at **Lite** license level.

If offline editing and uploading edits to a sync-enabled feature service are required the app must be licensed at the **Basic** license level.

Note:
See License your app for more information on license levels.

Copyright © 1995-2020 Esri. All rights reserved.

Utility networks

An example of an electric system represented as a utility network in an ArcGIS Runtime app

With ArcGIS Runtime you can add the following utility network capabilities to your app:

- Display a utility network(section below)—Visualize assets on a map and see how assets are connected.
- Trace how resources (gas, water, electricity, and so on) flow through a utility system network. For details, see this guide's trace a utility network topic. To perform traces, your app must be licensed with a Utility Network extension license.
- Get associated utility elements. Associations let you model connectivity between points that are not coincident, the structural support of assets, and features encased within other features. For details, see the association section in this guide's Trace a utility network topic.

To create, edit, manage, and configure a utility network, use ArcGIS Pro. Create it within an enterprise geodatabase, configure it, and publish it as a feature service so that it can be used in your ArcGIS Runtime app. For more information on creating utility networks, see in the utility network creation and configuration in the ArcGIS Pro documentation.

Every utility network contains at least one domain network and may contain tiers, subnetworks, and subnetwork controllers. The properties of these components define the characteristics of the subnetworks and drive tracing operations. In your ArcGIS Runtime app, you can access schema information about these components from a utility network definition (object `UtilityNetworkDefinition`) .

For more information about utility networks, see What is a utility network in ArcGIS Pro's help.

For information about the logical structure of a utility network, see Structure of a utility network in ArcGIS Pro's help. Note that feature classes in ArcGIS Pro are similar to feature tables in ArcGIS Runtime.

Display a utility network

You can display and share a complete utility network using a web map. You should include at most one feature layer for every feature table in the utility network, as described in Access the utility network.

As an alternative to using a web map, you can create your own map programmatically, making use of subtype feature layers (known as subtype group layers in ArcGIS Pro and in the web map). Subtype feature layers allow symbology and other layer properties to be set on a per-subtype basis. Utility networks make extensive use of subtypes (referred to as asset groups within a utility network). For example, a device feature table from an electric utility network would include subtypes for fuses, switches, transformers, and other kinds of equipment.

Copyright © 1995-2020 Esri. All rights reserved.

Geometries

Geometries represent real-world objects by defining a shape at a specific geographic location. They are used throughout the API to represent the shapes of features and graphics, layer extents, viewpoints, and GPS locations. They are also used as inputs and outputs of spatial analysis and geoprocessing operations, and to measure distances and areas, among other uses.

The `Geometry` class provides functionality common to all types of geometry. `MapPoint`, `Multipoint`, `Polyline`, `Polygon`, and `Envelope` all inherit from `Geometry`, and represent different types of shapes.

The following are common geometry characteristics:

- Geometries have a spatial reference indicating the coordinate system used by its coordinates.
- Geometries can be empty, indicating that they have no specific location or shape.
- Geometries can have z-values and/or m-values.
- Geometries can be converted to and from JSON to be persisted or to be exchanged directly with REST services.

Geometries are immutable

Most geometries are created and not changed for their lifetime. Examples include features created to be stored in a geodatabase or read from a non-editable layer, and features returned from tasks such as a spatial query, geocode operation, network trace, or geoprocessing task. Immutable geometries (geometries that cannot be changed) offer some important benefits to your app. They are inherently thread-safe, they help prevent inadvertent changes, and they allow for certain performance optimizations.

While immutable geometries appear to present problems for editing existing geometries, those problems are solved by using geometry builders. Geometry builders are designed to represent the state of a geometry under construction while allowing modifications, thus enabling editing workflows. See Edit geometriesfor more information.

Envelope

`Envelope` geometries represent rectangular shapes with sides that are parallel with the x or y axis of the coordinate system. Most commonly, they represent the spatial extent of layers or other geometries, or define areas of interest for tasks. They can be used as the geometry of graphics and in many geometry operations, although they cannot be used as the geometry of features.

New instances of `Envelope` are defined by specifying a minimum and maximum x-coordinate and minimum and maximum y-coordinate, and a `SpatialReference`. Optionally, a minimum and maximum z-value can be specified to define the depth of the envelope.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Envelope with bounding coords defined in longitude and latitude
var areaOfInterest = new Envelope(-117.85, 32.20, -116.50, 33.45, SpatialReferences.Wgs84);
```

Another way to define an envelope is by using two points. The minimum and maximum x,y coordinates are calculated from the two points.

```
// Envelope with lower-left (southwest) and upper-right (northeast) points
var southwestPoint = new MapPoint(-117.85, 32.20, SpatialReferences.Wgs84);
var northeastPoint = new MapPoint(-116.50, 33.45, SpatialReferences.Wgs84);
var env = new Envelope(southwestPoint, northeastPoint);
```

Point

`MapPoint` geometries represent a single point, place, or location such as a geocoded house address in a neighborhood or the location of a water meter in a water utility network. Larger geographic entities such as cities can be represented as points on small-scale maps. Points can be used as the geometry of features and graphics and are often used to help construct other geometries. They are also used in `Viewpoints`.

Points store a single set of x,y coordinates that represent the coordinates of a location (longitude and latitude for example), and a `SpatialReference`. Optionally, a z-value (representing things such as elevation for example) can also be included. Instances of `MapPoint` can be created using constructors, which define the full geometry in a single call.

```
// create a point with x, y, and z coordinate values using the WGS 1984 coordinate system
var saltLakeCity = new MapPoint(-111.88, 40.75, 1320, SpatialReferences.Wgs84);
```

⚠ **Caution:**

Remember when working with `MapPoint` instances that have a geographic spatial reference, the x-coordinate is the longitude (east or west), and the y-coordinate is the latitude (north or south). When geographic coordinates are represented in strings, they are generally written using the form "(latitude, longitude)", where the y-coordinate comes before the x-coordinate. Latitude values south of the equator and longitude values west of the prime meridian are expressed as negative numbers.

You can use `CoordinateFormatter` to convert a latitude, longitude formatted string directly to a `MapPoint`, and also return a latitude, longitude formatted string from an existing `MapPoint`. Other coordinate notations, such as Military Grid Reference System (MGRS) and United States National Grid (USNG) are also supported.

Multipoint

`Multipoint` geometries represent an ordered collection of points. They can be used as the geometry of features and graphics, or as input or output of geometry operations. For features that consist of a very large number of points that share the same set of attribute values, multipoints may be more efficient to store and analyze in a geodatabase compared to using multiple point features.

Multipoints are composed of a single read-only collection of `MapPoints`. You can create a new `Multipoint` by calling a constructor and passing in an iterable set of `MapPoints`.

```
var pointCollection = new MapPoint[] { southwestPoint, northeastPoint, saltLakeCity };
var oneMultipoint = new Multipoint(pointCollection, SpatialReferences.Wgs84);
```

Copyright © 1995-2020 Esri. All rights reserved.

To access each `MapPoint` in an existing `Multipoint`, iterate over the `ReadOnlyPointCollection` returned from `Points`.

Use the `MultipointBuilder` to build a multipoint one point at a time or to modify an existing multipoint.

Polyline

`Polyline` geometries represent the shape and location of linear features, for example, a street in a road network or a pipeline in an oil refinery. They can be used as the geometry of features and graphics, or as input or output of tasks or geoprocessing operations, such as the output of a network trace.

Polylines are composed of a series of connected segments, where each segment defines a continuous line between a start and an end point. You can also work with polylines by using point-based helper methods.

For example, use a constructor to create a new `Polyline` from a collection of `MapPoints`. This creates a series of straight `LineSegments` connecting the points you specified.

```
// define some map points (airports: LA, Chicago, Paris)
var laxPoint = new MapPoint(-118.408, 33.943, SpatialReferences.Wgs84);
var ordPoint = new MapPoint(-87.905, 41.979, SpatialReferences.Wgs84);
var orlyPoint = new MapPoint(2.379, 48.723, SpatialReferences.Wgs84);

// add the points to a collection
var stops = new MapPoint[] { laxPoint, ordPoint, orlyPoint };

// create a new Polyline from the points
var myFlightPath = new Esri.ArcGISRuntime.Geometry.Polyline(stops);
```

Polylines can have multiple parts. Each part is a series of connected segments, but the parts can be disjoint from each other, for example, in a polyline representing a discontinuous highway that has an unfinished section. Parts can also intersect at one or more vertices, for example, in a polyline representing a river and its tributaries. The `Polyline` class inherits from Multipart, which provides members for iterating through the segments and points of each part in a `Polyline`.

To build a polyline one point at a time, or tomodify an existing polyline, use `PolylineBuilder`. See Multipart geometries.

Polygon

`Polygon` geometries represent the shape and location of areas, for example, a country or a lake. They can be used as the geometry of features and graphics, or as input or output of tasks or geoprocessing operations, such as the output of a drive-time analysis or a buffer operation.

Polygons are similar to polylines in that they are also composed of a series of connected segments. However, polygons define closed areas, so the end point of the last segment is always in the same location as the start point of the first segment, forming a closed boundary. As with polylines, you can work with the vertices of the segments of a polygon by using point-based helper methods.

For example, use a constructor to create a new `Polygon` from a collection of `MapPoints`. This creates a series of straight `LineSegments` connecting the points you specified.

```
// create a point collection; add each corner of the state of Colorado
var coloradoCorners = new Esri.ArcGISRuntime.Geometry.PointCollection(SpatialReferences.Wgs84);
coloradoCorners.Add(-109.048, 40.998);
coloradoCorners.Add(-102.047, 40.998);
coloradoCorners.Add(-102.037, 36.989);
coloradoCorners.Add(-109.048, 36.998);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// create a new polygon from the point collection
var coloradoPoly = new Esri.ArcGISRuntime.Geometry.Polygon(coloradoCorners);
```

Tip:

When defining a polygon, there is no need to explicitly close it by repeating the start point as the last point. Polygons are always drawn as enclosed areas by ArcGIS Runtime. However, you may need to simplify a geometry to store it in a geodatabase.

To build a polygon one point at a time, or modify an existing polygon, use a `PolygonBuilder`. See Multipart geometries.

Similar to polylines, polygons can have multiple parts but have different rules than those for multipart polylines. Each part of a multipart polygon is a series of connected segments forming a closed ring. Each part must not cross any other part but may lie completely inside or outside another part. For example, a polygon representing the state of Hawaii would comprise eight disjoint parts, one representing each island. A polygon representing the country of South Africa, which completely surrounds the enclave of Lesotho, would comprise two parts, one contained inside the other. The `Polygon` class also inherits from Multipart.

Multipart (Polygon and Polyline)

`Polygon` and `Polyline` inherit from `Multipart`, which in turn inherits from `Geometry`. Multipart provides access to the geometry's `ReadOnlyPartCollection`. Each `ReadOnlyPart` in the collection is a collection of `Segment` objects. You can iterate through the segments or points in each part.

```
// loop through all parts in a Polygon
foreach (var island in hawaiiPoly.Parts)
{
    // loop through the points of each part
    var points = island.Points;
    foreach (var pt in points)
    {
        // read the point coordinates
        var coordValues = string.Format("x: {0} y: {1} z: {2}", pt.X, pt.Y, pt.Z);
        Debug.WriteLine(coordValues);
    }
}
```

A point-based convenience method is available to iterate through the `MapPoints` that represent the vertices of all the parts, in a single step.

```
// loop through all points (vertices) in all parts of a Polygon
foreach (var part in hawaiiPoly.Parts)
{
    foreach (var pt in part.Points)
    {
        // read the point coordinates
        var coordValues = string.Format("x: {0} y: {1} z: {2}", pt.X, pt.Y, pt.Z);
        Debug.WriteLine(coordValues);
    }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

In the same way as `Polygons` and `Polylines` are immutable, the collections returned from `Multipart` are also immutable—`ReadOnlyPartCollection`, `ReadOnlyPart`, and `ReadOnlyPointCollection`. However, when creating polygons and polylines, you use the mutable equivalents—`PartCollection`, `Part`, and `PointCollection`. Point-based helper methods are available on both the mutable and immutable classes.

Segments

A segment describes a continuous line between a start location and an end location. Every part in a multipart geometry is a collection of `Segments` where the end of one segment is at exactly the same location as the start of the following segment. The ArcGIS system supports both straight and curved segments, but at the current release, only straight `LineSegments` are supported in ArcGIS Runtime SDK. Multipart geometries can be composed from and decomposed into segments if required; until true curves are supported fully, using point-based methods offers equal functionality.

Because a single location is shared by adjacent segments, a single `MapPoint` object is used to represent the shared location when you iterate through the points in a part. As a result, when iterating through the points in a part of a polyline, there will be one more `MapPoint` than the number of `Segments` in that same part.

Similar to the geometries they comprise, `Segments` are immutable.

Other aspects of geometries

Spatial references

The meaning of the coordinates in a geometry is determined by the geometry's spatial reference. The vertices and spatial reference together allow your app to translate a real-world object from its location on the Earth to its location on your map or scene.

In some cases, a geometry's spatial reference may not be set. Graphics that do not have a spatial reference are drawn using the same spatial reference as the `MapView` in which they are drawn. When using a geometry builder to create a polyline or polygon geometry from point geometries, you don't need to set the spatial reference of every point before you add it to the builder, as it will be assigned the spatial reference of the builder it's added to. In most other cases, such as when using a geometry in geometry operations or when editing a feature table, the geometry's spatial reference must be set.

Learn more about spatial references

Linear, angular, and area units

Different types of unit of measurement can be used throughout the Runtime. Projected coordinate systems define coordinates using linear measurements, for example using meters or miles, which are represented by `LinearUnit`. Linear units are also used to return distance measurements, for example by some members of `GeometryEngine`. Geographic coordinate systems define coordinates using angular measurements, for example using degrees or radians, which are represented by `AngularUnit`. Methods that calculate the size of areas, for example in acres or square kilometers, use area units. These are represented by `AreaUnit`. Linear, angular, and area units can be defined by using enumerations of the most common units of measurement. They can also be defined by Well-Known ID (WKID) or Well-Known Text (WKT), which are listed in Coordinate systems and transformations.

Copyright © 1995-2020 Esri. All rights reserved.

Projection, topological, and other operations

Changing the coordinates of a geometry to have the same shape and location represented using a different spatial reference is known as "projection" or sometimes as "reprojection". Because geometries are immutable, they do not have any member methods that project, transform, or otherwise modify their content.

The `GeometryEngine` class provides a wide range of methods that read the content of geometries and modify that content to create new geometries. There are methods to project, rotate, move, cut, densify, and generalize geometries. When creating geometries using locations from the screen, and the map is a wraparound map, you may also need to consider normalizing a geometry before you save it to a geodatabase or use it in tasks or other operations.

Converting to and from JSON

Geometries can be serialized and de-serialized to and from JSON. The ArcGIS REST API documentation describes the JSON representation of geometry objects. You can use this encoding and decoding mechanism to exchange geometries with REST Web services or to store them in text files.

```
// return a polygon as json-formatted text
var jsonString = coloradoPoly.ToJson();
// re-create the polygon from json using the static FromJson method on Geometry
var newPoly = Esri.ArcGISRuntime.Geometry.Geometry.FromJson(jsonString);
```

Converting to or from JSON can add points to multipart geometries if the geometry has insufficient points to construct a segment.

Z-values and 3D

Geometries can have z-values, indicating values along the z-axis, which is orthogonal to both the x-axis and y-axis. Z-values can indicate height above or depth below a surface, or an absolute elevation. For example, z-values are used to draw the locations of geometries in `SceneViews`. Note that geometries are not considered true 3D shapes and are draped onto surfaces in the view, or in some cases, drawn in a single plane by using z-values. Z-values are stored on `MapPoints` and `Envelopes`. Therefore, because `Multipoints`, `Polylines`, and `Polygons` are created from `MapPoints`, all types of geometry can have z-values.

Whether or not a geometry has z-values is determined when the geometry is created; if you use a method that has a z-value parameter, the new geometry will have z-values (the geometry's `HasZ` will be true). If you create geometries using constructors that take z-value parameters, or if you pass into the constructor points or segments that have z-values, the new geometry will have z-values. A geometry with z-values is sometimes known as a z-aware geometry.

It may be that not all vertices in your geometry have a z-value defined. `NaN` is a valid z-value used to indicate an unknown z-value. However, the default z-value is zero. It is important to note that when you get z-values from a geometry that does not have z-values, the default value of zero is returned. Check the `HasZ` property to determine whether a z-value of zero means that there are no z-values in the geometry or that the z-value in the geometry's coordinates really is zero.

See Editing geometries for more information about editing workflows with respect to z-values.

M-values

M-values are used in linear referencing scenarios, and like z-values, every geometry can optionally store m-values. The default m-value is `NaN`. If an m-value is specified as a parameter when a geometry is created, the new geometry will have m-values (the geometry's `HasM` will be true). Note that when you get m-values back from a geometry, the default value of `NaN` (different than

Copyright © 1995-2020 Esri. All rights reserved.

the default for z-values) is returned for vertices that do not have m-values. A geometry with m-values is sometimes known as an m-aware geometry. See Editing geometries for more information about editing workflows and m-values.

Related topics

Spatial references
Edit geometries

Copyright © 1995-2020 Esri. All rights reserved.

Spatial references

A spatial reference is a characteristic of a geometry that identifies how its coordinates relate to real-world space. It is important for ensuring that spatial data in different layers, graphic overlays, and feature sets can be used together for accurate viewing or analysis.

Spatial references can be referred to by a well-known ID (WKID)—an integer value. Some common WKIDs are mentioned in the text below; for a more complete description, see Spatial reference specifications in this topic.

NOT IMPLEMENTED

Why spatial references are important

To integrate spatial data together into a map or when performing analysis, ArcGIS Runtime must know where things are located on the Earth's surface and it uses coordinates to do this. Coordinates are expressed with respect to a coordinate system, which is a frame of reference around a model of the Earth's surface. Not all coordinates and their associated coordinate systems are the same; they can use various units (degrees minutes seconds, decimal degrees, or meters) and they can be based on different types of models. ArcGIS Runtime uses mathematical transformations to reproject coordinates from one coordinate system to another. A spatial reference provides all the information needed for reprojection.

Coordinate systems and projections

Coordinate systems are split into categories: geographic, projected, and local:

- Geographic coordinate systems (GCSs) use a three-dimensional ellipsoidal surface to define locations. The coordinates are based on angles from the center of the Earth to the surface. Typically GCSs use latitude and longitude specified in degrees. The coordinates derived from a GPS device are returned in a GCS named WGS84 (WKID=4326).
- Projected coordinate systems (PCSs) are variously described as planar (two-dimensional), Cartesian, or "flat." Unlike a GCS, a PCS has constant lengths, angles, and areas across the two dimensions. PCSs use a geographic coordinate system projected onto a flat surface for display. There are various projections with different desirable characteristics. Some preserve accuracy in particular areas of the Earth, others are better at maintaining the shape of features, while others favor accurate area or distance measurements. Coordinates are identified by x,y coordinates on a grid. Most basemaps from ArcGIS Online, Google, and OpenStreetMap use the same projected coordinate system named Web Mercator Auxiliary Sphere (WKID=3857).
- Local coordinate systems are often unrelated to any other coordinate system. The origin and the x,y coordinate units correspond to a local point of reference. Because the relationship of a local coordinate system to another coordinate system cannot be established easily, these are sometimes referred to as unknown coordinate systems.

When you need to know about spatial references

When you use layers with different spatial references for viewing or analysis, ArcGIS Runtime automatically reprojects geometries or requesting data in the appropriate spatial reference from services where possible and appropriate. Nevertheless, occasionally you'll need to know about how spatial references are used.

Add layers to the map

When you create a map, the spatial reference of the first layer you add is used as the spatial reference used by the entire map; this is typically the basemap. When ArcGIS Runtime renders a map, it draws all the data using the same spatial reference so that the data lines up properly. If a group layer is the first layer added to a map, the map will use the spatial reference of the first layer in the group layer. If that layer is also a group layer, the group layer will be searched recursively until a non-group layer is found.

Copyright © 1995-2020 Esri. All rights reserved.

As you add additional layers to your map, you may need to request those layers from the service in the same spatial reference your map is using. Whether or not you must request this depends on what type of layer you're adding. The following sections describe layer types as they relate to setting your map's spatial reference.

Graphics overlays

Graphics overlays support on-the-fly reprojection of graphics. On-the-fly reprojection requires extra processing that could slow down the map view rendering time. When you add a graphic to a graphics overlay, it is best if the geometry in the graphic has the same spatial reference as the map. You can explicitly convert geometries to the required spatial reference.

Feature layers from feature services

When using a feature service table created from a feature service from ArcGIS Online or ArcGIS Enterprise, the service supports reprojection. When initialized, the table and its associated feature layer will automatically be set to the same spatial reference as the map. This ensures the data is requested from the service in the correct coordinates, without the need to explicitly define the spatial reference for the feature table. ArcGIS Runtime determines the correct spatial reference and requests data from the feature service accordingly.

Feature layers from geodatabase feature tables

Your tables in the geodatabase do not need be in the same spatial references as the map you are adding them to, because on-the-fly reprojection of data from these tables is supported. However reprojection comes with a cost in drawing performance, and avoiding reprojection can help maximize performance. To control the spatial reference of your tables, ensure the ArcMap map frame is using the desired spatial reference before you run the create runtime content tool. If you're using the services workflow, set the desired spatial reference in the parameters used to generate the geodatabase. For details on the desktop and services workflows, see Work with offline maps.

Dynamic map service layers

If these are ArcGIS Online or ArcGIS Enterprise map services, then the server supports reprojection on the fly. When you add an ArcGIS dynamic map service layer to the map, ArcGIS Runtime automatically asks for the map image in the correct spatial reference for you (based on the map's spatial reference).

Tiled layers

Tiled layers are cached layers. At the time of caching, a spatial reference is used and is therefore predefined for the tiled layer. It's typically not possible to request tiled layers in a different spatial reference from the one defined in the service using that cache (unless the server supports doing this on the fly; most do not). If an ArcGIS tiled layer is added to a map with a different spatial reference from the tiled layer, it cannot be drawn.

Raster layers

Raster layers support on-the-fly datum transformation and reprojection. For more information on using raster layers in your app, see Add raster data.

Edit data

When creating new features from coordinates, the coordinate values must match the spatial reference for the layer; otherwise, the features will not show up in the correct location on the map.

Copyright © 1995-2020 Esri. All rights reserved.

Perform analysis

Geometry objects used for analysis (for example, determining spatial relationships, such as where polygons intersect) require that the spatial reference be known before the analysis is performed. Otherwise, results may not be accurate. Likewise, it's meaningless to compare two geometries or determine their spatial relationship if they have different spatial references. To display a geometry in a map layer, the geometry must have either the same spatial reference as the layer or be projected to the layer's spatial reference. To use two geometries together, they should have the same spatial reference.

Convert geometries from one spatial reference to another

When using the geometry engine to convert geometries from one spatial reference to another, the source and destination spatial references must be specified. For details, see Projection, topological, and other operations in the Geometries topic. An appropriate datum transformation is used by default. You can also specify the transformation you want to use. You can also convert strings containing coordinates formatted as latitude and longitude directly to points, and vice-versa. Other types of coordinates such as Universal Transverse Mercator (UTM) and United States National Grid (USNG) are also supported.

Spatial reference specifications

To define a spatial reference, you can use either a Well-Known ID (WKID) integer value or a text string definition referred to as Well-Known Text (WKT). WKIDs are defined by standards bodies or organizations, with each value representing a specific spatial reference. ArcGIS supports a variety of WKIDs, typically those defined by the European Petroleum Survey Group (EPSG) or Esri, as well as a few other commonly used IDs. In contrast, WKT text describes of all of the parameters of a spatial reference. To see a list of supported WKIDs and their WKT definitions for geographic coordinate systems, projected coordinate systems, and transformations, see Coordinate systems and transformations.

Linear, angular, and area units can also be defined by WKID or WKT, and are used by many spatial reference and geometry related API members. These WKIDs are also included in Coordinate systems and transformations.

Geographic transformations

Geographic transformations are functions used to transform coordinates between spatial references that have different datums. The transformations are used when geometries must be projected from one spatial reference to another when there is a difference in the datum that underlies the two spatial references. Using the most suitable transformation ensures the best possible accuracy for the reprojection. Different transformations are suitable for different geographical areas.

Esri's Projection Engine is used for geographic transformations and supports many predefined geographic transformations. There is not a defined transformation from every spatial reference to every other—such a collection would be very large. Therefore, each geographic transformation is usually composed of two steps, each step defining a datum transformation and the inverse of another datum transformation. Each geographic transformation step can be constructed from a well-known ID (WKID) that represents a datum transformation. The list of supported WKIDs (in PDF format at Geographic transformations) includes a transformation from every supported datum to the datum named World Geodetic System 1984 (WGS84). Geographic transformations with more than one step typically go via WGS84, with one forward and one inverse datum transformation chained together to perform the complete geographic transformation. Additionally, there is limited list of datum transformations directly between two non-WGS84 datums, such as WKID `4461`, named `NAD_1983_HARN_To_NAD_1983_NSRS2007_1`. These may be used to create one-step geographic transformations.

Datum transformations can be mathematically defined by specific equations (equation-based transformations), or may rely on external supporting files (grid-based transformations). Exactly how coordinates should be transformed is defined by agencies such as the US National Geodetic Survey, and then identified by a WKID integer number and WKT text string definition, similarly to how spatial references are identified.

Copyright © 1995-2020 Esri. All rights reserved.

As a developer, you can rely on ArcGIS Runtime to choose a default geographic transformation for you, resulting in accurate coordinate reprojection. Or in some situations, you may use a geographic transformation of your own choosing. You may specify a geographic transformation each time you use the geometry engine (the `GeometryEngine` class) to project a geometry, or you can customize the default geographic transformation used by your app through a JSON configuration file. You can even choose from a list of suitable transformations from the transformation catalog (the `TransformationCatalog` class) for a given pair of spatial references and optionally a given geographic area, and choose one of those. If you have even more specialized needs, you can create a custom transformation using WKT.

Grid-based transformations

Datum transformations can be mathematically defined (equation-based transformations), or may rely on external supporting files (grid-based transformations). Certain Projection Engine data files must be present when you use a grid-based transformation in your app; attempting to use a transformation with missing Projection Engine files will cause an error. The API can detect whether the necessary files are available on the local file system.

Note:
These Projection Engine data files are available for download from the downloads page on developers.arcgis.com (requires you to log in).

The data files should reside in a location specified in the transformation catalog's Projection Engine location. There is no default location set by this API. If you set the Projection Engine location , you must do so early in your app before using other API calls that use the Projection Engine.

The following function takes a folder path (string) and if it represents a valid directory on the local file system, assigns it as the projection engine directory. If you needed specific transformations, you could also check for the required Projection Engine files in the specified location.

```
private void SetProjEnginePath(string peFolderPath)
{
    // make sure the directory exists before assigning it
    if (System.IO.Directory.Exists(peFolderPath))
    {
        // assign the path to the static ProjectionEngineDirectory property
        TransformationCatalog.ProjectionEngineDirectory = peFolderPath;
    }
}
```

Use default transformations

A common use case is to transform a geometry, creating a new geometry that uses the same spatial reference as the map. The geometry engine's `Project` method internally uses the best overall geographic transformation available by default without you needing to specify one. Using the transformation catalog you can find out which geographic transformation is used by default when projecting between any two spatial references.

If no transformation is required for the two spatial references (for example, two projected coordinate systems that have the same underlying geographic coordinate system), `getTransformation` will return `null`.

The following function takes WKIDs for an input and output spatial reference and returns the default transformation used to transform between them.

Copyright © 1995-2020 Esri. All rights reserved.

```
private DatumTransformation GetDefaultTransformation(int wkidIn, int wkidOut)
{
    // use the provided WKIDs to create the input and output spatial references
    SpatialReference srIn = SpatialReference.Create(wkidIn);
    SpatialReference srOut = SpatialReference.Create(wkidOut);

    // call the static GetTransformation method with the in/out spatial references to get the
default transformation
    DatumTransformation defaultTransformation = TransformationCatalog.GetTransformation(srIn,
srOut);

    return defaultTransformation;
}
```

Customize default transformations

There may be cases where you want to specify your own default geographic transformations. This would happen when the defaults chosen by ArcGIS Runtime are not suitable to your needs but you do not want to write code to specify the geographic transformation every time you need to use a one. To change any defaults, create a JSON file named `gtdefaults_overrides.json` and place it in the location returned from the transformation catalog method `projectionEngineLocation`.

Note:
You must set `projectionEngineLocation` early in your app before using other API calls that use the Projection Engine. There is no default location set by this API.

Note:
Changing the JSON file will have no effect on running apps. To use this file in your app, you must change it before running your app.

The file's content looks like the following:

```
{ "geogtran" : [
   [   3819,   3906,   3817,    0,   3962,    1 ],
   [   3819,   4075,   3817,    0,   4077,    1 ],
   [   3819,   4156,   3817,    0,  15965,    1 ],
   [   3819,   4178,   3817,    0,  15996,    1 ],
   [   3819,   4179,   3817,    0,  15995,    1 ],
   [ 104248, 104896, 108038,    0,      0,    0 ],
   [ 104257, 104896, 108019,    1,      0,    0 ]
]}
```

This file contains one line per datum pair ("from" and "to"). Only include lines for the default geographic transformations that you want to override. The six columns have the following information.

1. The WKID of the "from" datum.
2. The WKID of the "to" datum.
3. The WKID of the first datum transformation.
4. Whether to use the first datum transformation as-is (`0`) or use its inverse (`1`).
5. The WKID of the second datum transformations WKID (or `0` if this is a single-step geographic transformation).

Copyright © 1995-2020 Esri. All rights reserved.

6. Whether to use the second datum transformation as-is (`0`) or use its inverse (`1`). (`0` if this is a single-step geographic transformation).

ArcGIS Runtime checks whether a geographic transformation defined in this file is useable before attempting to use it. If it is not, the geographic transformation for that datum pair uses the previous default.

Choose a transformation from a list

From the transformation catalog, you can retrieve a list of geographic transformations useable between two spatial references. The `TransformationsBySuitability` method returns a list of geographic transformations in descending order by suitability. With this, you could provide your app users with a workflow wherein they are able to choose from a list of suitable transformations.

```
private IReadOnlyList<DatumTransformation> FindSuitableTransformations(int wkidIn, int wkidOut,
Envelope aoi)
{
    // use the provided WKIDs to create the input and output spatial references
    SpatialReference srIn = SpatialReference.Create(wkidIn);
    SpatialReference srOut = SpatialReference.Create(wkidOut);

    // call the static GetTransformationsBySuitability method with the in/out spatial references
and area of interest
    // the list returned is sorted in descending order of suitability (first is most suitable,
ie)
    IReadOnlyList<DatumTransformation> bestTransforms =
TransformationCatalog.GetTransformationsBySuitability(srIn, srOut, aoi);

    return bestTransforms;
}
```

The list returned from `TransformationsBySuitability` may include grid-based transformations whose supporting files are not on the local file system. You can identify these cases using the `IsMissingProjectionEngineFiles` method on the geographic transformation step object (the `GeographicTransformationStep` class). Your app could go so far as informing the user about precisely which files are missing, or even automatically download the missing files from a known location. Retrieve the list of required files using the `ProjectionEngineFilenames` method of the geographic transformation step object.

Consider the work area extent

Your work area may be small and located where the best geographic transformation for the entire spatial reference is not the best for your work area. You may be able to improve accuracy by finding the best geographic transformation for your specific work area. Use the overloads of the transformation catalog's `Transformation` and `TransformationsBySuitability` methods to pass in the extent for which you'll be transforming geometries.

Create a custom transformation

Some users adjust the components of a geographic transformation to suit their specific needs. These custom transformations use the existing methods of transformation (`Position_Vector`, `Geocentric_Translation`, and so on) but have different parameter values than the well-known definitions. Create a custom geographic transformation object by passing in a text string containing your custom transformation expressed in WKT format. The resulting geographic transformation will have a WKID of

Copyright © 1995-2020 Esri. All rights reserved.

zero. Before using a custom transformation that requires Projection Engine files, make sure that those files are present and the Projection Engine location has been set.

Related topics

Geometries
Edit geometries
Coordinate systems and transformations

Copyright © 1995-2020 Esri. All rights reserved.

Local Server

Local services allow you to deploy services that run locally with your app to provide functionality like geocoding, geoprocessing, routing, and so on. In ArcGIS Runtime SDK for .NET, local services are only available for Windows Presentation Framework (WPF) apps. An app built for another ArcGIS Runtime for .NET platform can use such functionality from an online service, but not locally.

This topic does not apply for the iOS platform.

Copyright © 1995-2020 Esri. All rights reserved.

Work with maps (2D)

Copyright © 1995-2020 Esri. All rights reserved.

Display a map

Maps specify how geographic data is organized and how it should be communicated to your app users. Each instance of a map (`Map`) represents an individual map you can display for your users. To display a map you must assign it to a map view (`MapView`). In an MVC architecture, the map represents the model tier and the map view represents the view tier. The map and map view work together to visualize the geographic data on a screen.

Users can interact with the map and its layers using the map view's built-in navigation tools.

Open a map

You can open a map by creating it from a web map or from a mobile map package.

Create the map object from a web map

Web maps can be accessed using either their URL or directly from a portal using their portal item ID. Both methods are described here along with the authentication process required to access the map if it or its data is not public.

Using a web map URL

You can construct the map object from one of the map's URL strings as follows:

```
// use a static method on Map to create a map from a web map url
var webMap = await Map.LoadFromUriAsync(new Uri(UrlTextBox.Text));

// add the map to the map view
MyMapView.Map = webMap;
```

The map constructor supports these three URL formats:

- The map viewer URL—For example, `https://www.arcgis.com/home/webmap/viewer.html?webmap=69fdcd8e40734712aaec34194d4b988c`. The URL is provided in the browser's address bar when a web map is created or viewed using the ArcGIS.com map viewer.
- The item details URL—For example, `https://www.arcgis.com/home/item.html?id=69fdcd8e40734712aaec34194d4b988c`. This URL is found in the address bar when you're viewing the details page for an item presented in **My Content** in ArcGIS.com.
- The item data URL—For example, `https://www.arcgis.com/sharing/rest/content/items/69fdcd8e40734712aaec34194d4b988c/data?`. This is the address of the JSON string that represents the information to display the map.

If the map is not public, the API's authentication challenge handler automatically requests that the user provides his credentials.

Using a web map portal item

You can construct the map object using the map's portal item ID. You can find this 32 character ID within any of the three URLs described above. For example, here is a map that's identified by the item ID of `69fdcd8e40734712aaec34194d4b988c`.

Copyright © 1995-2020 Esri. All rights reserved.

1. Create the portal object by providing the URL string to the portal. In the example below, the public map is accessed from ArcGIS Online (www.arcgis.com).

```
// connect to ArcGIS Online (default portal if no URL is passed to ArcGISPortal.CreateAsync)
ArcGISPortal arcGISOnline = await ArcGISPortal.CreateAsync();
```

2. Construct the portal item object by providing the portal and the item ID string for the map.

```
// create an PortalItem using a portal item ID provided by the user
var portalItem = await PortalItem.CreateAsync(arcGISOnline, ItemIdTextBox.Text);
```

3. Pass the portal item to the map constructor.

```
// pass the portal item to the constructor for a new Map
var map = new Map(portalItem);
```

Upon completion of this code, neither the portal, portal item, nor the map are loaded. These objects will be loaded when you pass the map to the map view or if you load them explicitly. See the loadable pattern topic for more information. You can override any of the map's properties by setting their value before the map is loaded.

Authentication

If the map or its layers are not public, you need to provide credentials to access them. In these cases you can either rely on the API's authentication challenge handler to prompt the user for their credentials or you can provide your own login facility and pass the credentials to the portal's `Credential` property as described in the Access the ArcGIS platform topic.

Copyright © 1995-2020 Esri. All rights reserved.

Create a map object from a mobile map package

A mobile map package can be created with ArcGIS Pro or taken offline with the ArcGIS Runtime SDK (on-demand or preplanned). It is a transport mechanism for maps, their layers, data, networks, and locators that can be can be easily copied onto a device.

Note:
StreetMap Premium also provides ready-to-use and regularly updated mobile map packages that include a street map, locator, and network dataset for areas throughout the globe. For details, see Add StreetMap Premium data.

Each mobile map package can contain one or more maps that can be displayed in a map view. For more information, see Create mobile map package.

1. Obtain the file path to the mobile map package.
```
var mmpkPath = Path.Combine(NSBundle.MainBundle.BundlePath, "CaliforniaNevada.mmpk");
```
2. Construct the mobile map package object using the file path.
```
MobileMapPackage mmp = await MobileMapPackage.OpenAsync(mmpkPath);
```
3. The mobile map package must be loaded into the app before the maps themselves can be accessed. Once the mobile map package is loaded, any of its maps can be assigned to the map view to be displayed.
```
// get the first map in the package
var map = mapPackage.Maps[0];
```

Upon completion of this code the map is not loaded. You can override any of the map's properties by setting their value before the map is loaded. See the loadable pattern topic for more information.

Display the map in a map view

To display a map, you assign it to a map view. Each map view can display a different visible area of the map, can be configured in different ways, and can be designed to to user gestures. Assign a map to a map view as follows:

```
// assign a map to the map view
MyMapView.Map = map;
```

Assigning a map to a map view initiates the map's load cycle. When that cycle completes, the map view initiates the load cycle on the map's layers in order to draw them. For more information loading resources (load cycles), see Loadable pattern of asynchronous resources. You can monitor the loading of the individual layers using the map view's `LayerViewStateChanged` event handler .

You can load the map into your app before assigning it to a map view. This is useful if you want to validate the content of the map or you want to access any content or properties in the map without displaying it. For example, this code loads the map explicitly and checks if the map contains bookmarks before it displays the map in the map view.

```
await map.LoadAsync();
if(map.Bookmarks.Count > 0)
```

Copyright © 1995-2020 Esri. All rights reserved.

```
{
    // ... display bookmarks
}
```

Monitor layer loading

Assigning a map to a map view or explicitly loading a map triggers loading of the map's basemap and operational layers. Each time the state of a layer changes, the map view's `LayerViewStateChanged` event handler is triggered. Examine the layer's view status to determine whether a layer is active, loading, not visible, out of scale, or whether an error has occurred with the layer's loading process.

```
// handle the LayerViewStateChanged event for the map view
MyMapView.LayerViewStateChanged += (s, e) =>
{
    // get the name of the layer that raised the event
    var layerName = e.Layer.Name;

    // check the status and report the value
    switch (e.LayerViewState.Status)
    {
        case LayerViewStatus.Active:
            Console.WriteLine(layerName + " is active.");
            break;
        case LayerViewStatus.Error:
            Console.WriteLine("Error displaying " + layerName);
            break;
        case LayerViewStatus.Loading:
            Console.WriteLine(layerName + " is loading.");
            break;
        case LayerViewStatus.NotVisible:
            Console.WriteLine(layerName + " is not visible.");
            break;
        case LayerViewStatus.OutOfScale:
            Console.WriteLine(layerName + " is not displayed due to the current scale.");
            break;
        default:
            Console.WriteLine("Status of " + layerName + " is unknown.");
            break;
    }
};
```

Share data between maps and map views

Sharing the same map instance simultaneously among two or more map views, or a layer instance simultaneously between maps, is not supported and is likely to cause stability or performance issues. Working with graphics overlays, or distinct feature layer instances that share the same data source (feature table, for example), can cause the same issues and is also not supported in multiple map views at the same time.

If you need to reuse a map in another map view, you should ensure that it is first removed from the previous map view. The same technique can be used for reusing a layer instance between maps.

Monitor map drawing

If your users pan or zoom the map or the business logic in your app changes the visible area of the map, then new map content will need to be drawn. There will be a short time delay before this map drawing phase is completed. Your app may

Copyright © 1995-2020 Esri. All rights reserved.

need to know that the drawing is complete. For example, you may want your app to take a snapshot, or screenshot, of the map only when the drawing has finished.

Use the map view's `DrawStatusChanged` event to detect whether the map drawing is in progress or has completed. For example, this code displays an activity indicator if the drawing is still in progress.

You can use the `DrawStatusChanged` event to hide an activity indicator when drawing is complete.

```
// handle the DrawStatusChanged event for the map view
MyMapView.DrawStatusChanged += (s, info) =>
{
    // if drawing is complete, hide the activity indicator
    if(info.Status == Esri.ArcGISRuntime.UI.DrawStatus.Completed)
    {
        Console.WriteLine("Map view drawing is complete.");
        DrawProgress.Visibility = Visibility.Hidden;
    }
};
```

Navigate the map

When a map is first loaded into a map view, it's displayed at a geographical location defined by the map's initial view point. Users can then pan or zoom the map with a range of gestures that are built into the map view. You can access the visible area of the map as a polygon returned by the map view's `VisibleArea` property . The visible area is returned as a polygon, and not an envelope, because the map may be rotated and each corner of the map may contain unique x-y coordinates.

```
Esri.ArcGISRuntime.Geometry.Polygon visibleMapArea = MyMapView.VisibleArea;
```

As a developer you can programmatically set the map's visible area in your app by calling one of the map view's many `setViewpoint` methods. Many options are available to choose from to allow you to:

- Rotate the map to a specified angle.
- Zoom the map to the specified scale.
- Zoom or pan the map so that a given geometry fits the visible area of the map view.
- Zoom or pan the map to a specified location.
- Zoom or pan the map to a specific view point. You can define a view point using a:
  - Center and scale
  - Center, scale, and rotation
  - Latitude, longitude, and scale
  - Target extent
  - Target extent and rotation

The following example zooms the map display to a specified polygon (the state of Utah).

```
// zoom to the extent of a geometry (polygon), add 20 pixels of padding
await MyMapView.SetViewpointGeometryAsync(utahPolygon, 20);
```

Copyright © 1995-2020 Esri. All rights reserved.

Configure the map view

You can enhance your user's map experience by changing the background color, showing the device's location, rotating the map display, or making the world map seamless by enabling wrap-around.

Change the map background color

You can set a background color to display behind the map. This color will be displayed in transparent areas of the map as well as areas where there is no basemap or operational data. Before a map is added to a map view, a default background grid is displayed. If a map is added that does not have a background color defined, the map view background grid will continue to display behind areas of the map without data. If a background color is applied to the map, it is used as the background display and any subsequent changes to the map view's background grid will have no effect.

```
myMap.BackgroundColor = System.Drawing.Color.LightSlateGray;
```

Show the device's location (display the user's current location)

Many devices provide information about their location through Wi-Fi, cellular networks, or GPS. The device's operating system passes its location to the map view so you can display and track the device on a map. The map view always tries to get the most accurate location available but the reported location could be a best approximation based on signal strength, satellite positions, and other factors. By default, the map view displays the location with a blue, round symbol and a semi-transparent blue circle around this symbol to indicate the range of accuracy.

The display of the location on a map, the symbol used, the animation, and the auto pan behavior are managed by an instance of `LocationDisplay`. Each `MapView` has its own instance of a location display.

The following example shows the current location and sets up a handler for when the location changes.

```
// handle location change (show location info, search nearby, update distance to destination,
etc.)
MyMapView.LocationDisplay.LocationChanged += OnLocationChanged;
  // start showing the current location of the device
MyMapView.LocationDisplay.IsEnabled = true;
```

For details and additional options for displaying a user's location, such as displaying tracks or changing the symbol, see Show device location in this guide. Also see `LocationDisplay` in the API Reference.

Enable wrap around

Most flat representations of the earth extend up to 180 degrees east and west longitude and make it difficult to visualize areas that span across the 180th meridian. Enabling wrap around in the map view displays a 2D map extending seamlessly across the 180th meridian. For more information, see the wrap around maps topic.

The following example enables wrap around mode if it's supported for the current map.

Copyright © 1995-2020 Esri. All rights reserved.

```
// try to enable wraparound mode
MyMapView.WrapAroundMode = Esri.ArcGISRuntime.UI.WrapAroundMode.EnabledWhenSupported;
// wrap around mode may not supported for this map, show status in the UI
WraparoundEnabledCheckbox.IsChecked = MyMapView.IsWrapAroundEnabled;
```

Note:

If your map has wrap around enabled and you're working with map locations, map extents, or are editing or sketching geometries with a map, you must normalize the geometries.

Some maps do not support wrap around; maps that do support wrap around will automatically be displayed in wrap around

Rotate the map

You may not want your map to display in a North-South direction in your device. You can reorient your map in the map view by setting the viewpoint rotation. A positive rotation value rotates the map in a counterclockwise and a negative rotation value rotates the map in a clockwise direction.

```
// rotate the map 90 degrees to the west (counterclockwise)
await MyMapView.SetViewpointRotationAsync(90);
```

Filter data by time

Some layer types are time-aware, meaning they contain temporal data that can be filtered. Runtime has rich support for time-aware layers and data. To learn more, see Visualize and compare data over time.

Related topics

Maps and scenes
Build a new map
Wraparound maps
Save a map

Copyright © 1995-2020 Esri. All rights reserved.

Build a new map

A map gives you a structure for organizing your geographical data so your users can view, explore, and interact with it. This page describes how to design and build a map with code and then save it for later use by another app or by a different user.

Note:
You can also create a map for use in your ArcGIS Runtime app using ArcGIS Online's map viewer (how-to steps are in ArcGIS Online Help's Get started with maps page).

To build a map with code for your ArcGIS Runtime app, you use this API to create a map. In this API, the map is represented by the `Map` class that gives you all the methods and properties you need to set the background layer, add the operational layers, set the spatial reference, add bookmarks, and save the map. If you wish to display the map and allow users to interact with it then you pass the map to the `MapView` class.

Map design considerations

When designing a map you must:

- Choose a background for your map so that your users can orient themselves. Esri provides a number of standard basemaps that you can use as a background or you can use other providers, such as your organization's basemaps. Alternatively, you can construct a basemap in code.
- Ensure that you specify a spatial reference for your map that allows the layers to be aligned and displayed together. It will be more efficient if all of your operational layers have the same spatial reference as the map and basemap because the operational layers will not need to be reprojected.
- Start the map display in a specific and recognizable geographical area.
- Only show the data that your users need.
  - Set the minimum and maximum scale thresholds for layers so only useful information is displayed at each scale.
  - You may wish to set a map reference scale. The reference scale of the map is the scale at which a user should view the map for the feature symbols and text to appear at their authored size. By setting a reference scale, layers and labels will not resize based on changes of scale in the map view.

Construct the map

This API provides several types of map constructors. Your choice will depend upon whether you wish to specify the desired basemap, spatial reference or initial viewpoint when you create the map. There is the all-encompassing constructor that you can use to create a map with a standard Esri Imagery basemap that is centered on a latitude and longitude location and zoomed into a specific level of detail.

```
// create a map centered on Corsica
var myNewMap = new Map(
    BasemapType.Imagery, // use the Imagery basemap
    42.175131,           // latitude
    9.192313,            // longitude
    8                    // level of detail (lower number = smaller scale)
    );
```

In this example `BasemapType.Imagery` refers to the ArcGIS Online service available at the REST end point https://services.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer. This basemap, like all the standard Esri

Copyright © 1995-2020 Esri. All rights reserved.

basemaps, uses the Web Mercator spatial reference and this, in turn, determines the spatial reference of the map. The latitude and longitude values center the map at a particular location and the `levelOfDetail` determines the scale of data that is displayed. So, this constructor will display a map at a specified location and scale no matter what size of map is displayed to the user.

Other map constructors

There are a number of other map constructors that you can use if you want to set the spatial reference, the basemap or the initial viewpoint at a later point in time. Your choice depends on the workflow in your app. Use this constructor if you:

- Know the basemap but not the starting location

```
// create a map that uses the National Geographic base map
var myNewMap = new Map(Basemap.CreateNationalGeographic());
```

- Know the spatial reference but not the basemap

```
// create a map that uses the Everest Bangladesh spatial reference
var myNewMap = new Map(SpatialReference.Create(37202));
```

- Want an empty map

```
// create a map without a basemap, layers, or spatial reference
var myNewMap = new Map();
```

Create a basemap

A basemap is composed of a collection of base layers and reference layers. The base layer can give your map a recognizable background so that your operational layers are displayed in context. The reference layer can display useful text labels such as the city, street, river or mountain names.

Base layers are displayed at the bottom of a map, and reference layers are displayed at the top, with the map's operational layers sandwiched between them. The content of a basemap is typically static and does not change frequently so tiled layers make good basemap layers as they provide faster access to relatively static maps. (For more information see Layer types.) As discussed previously, ArcGIS Online provides a number of ready-to-use standard basemaps such as the World imagery, topographic, oceans, streets, and so on. Or, you can add basemaps from your organization or another public service. Also, you can create your own basemap by combining (mashing up) layers.

Standard ArcGIS Online basemaps

To use, instantiate a basemap, such as the ArcGIS Online World Streets basemap.

```
// create a new (empty) map
var myMap = new Map();
// create a streets basemap
var streetsBasemap = Basemap.CreateStreets();
// add the basemap to the map
myMap.Basemap = streetsBasemap;
```

Copyright © 1995-2020 Esri. All rights reserved.

A single layer

You can also instantiate a basemap with a single layer. For example, you could instantiate a tiled layer, construct a basemap from that layer and then pass that basemap to the map.

```
// create a new tiled layer
var imageryTiledLayer = new ArcGISTiledLayer(new Uri("http://services.arcgisonline.com/arcgis/
rest/services/World_Imagery/MapServer"));
// create a basemap from the tiled layer
var myBasemap = new Basemap(imageryTiledLayer);
// create a new map; set the basemap
var myMap = new Map();
myMap.Basemap = myBasemap;
```

Combining layers into one basemap

You can combine a number of layers to create a single basemap as long as the layers are capable of being displayed in the same spatial reference (that is, either a layer is already in the same spatial reference as the other layers or it can be re-projected to the spatial reference of the other layers). In the following code example, the first layer added to the set of basemap layers establishes the spatial reference of the basemap, making it **3857** (you can see this number if you enter the URI in the code sample into your browser's address box and look for "spatial reference" in the resulting page). The second basemap layer, being a map image layer, is requested from the map server in spatial reference set by the first layer, **3857**.

```
// create two layers
var imageTiledLayer = new ArcGISTiledLayer(new Uri("http://services.arcgisonline.com/arcgis/rest/
services/World_Imagery/MapServer"));
var censusLayer = new ArcGISMapImageLayer(new Uri("http://sampleserver6.arcgisonline.com/arcgis/
rest/services/Census/MapServer"));
  // create a new basemap; add the layers (BaseLayers property)
var myBasemap = new Basemap();
myBasemap.BaseLayers.Add(imageTiledLayer);
myBasemap.BaseLayers.Add(censusLayer);
  // create a new Map to display the basemap
var myMap = new Map(myBasemap);
```

Notes

In terms of the map's spatial reference, it's important to understand the following:

- If a map does not have a spatial reference, it will automatically be set by the map's first basemap.
- If you add a basemap to a map that already has a spatial reference, it must have the same spatial reference as the map; otherwise it will fail to draw. The same is true if you exchange one basemap for another.
- If the map does not have a basemap, then the map will adopt the spatial reference of the first operational layer that is added to the map.

Just like the map, the basemap loads asynchronously. This is described in more detail in the loadable pattern topic.

A group layer cannot be included in a basemap.

Add operational layers

Operational layers are a category of layers that provide geographic content that is of specific interest to the app. Your choice of layer type will determine whether you can edit, query, identify or overwrite the layer rendering. For more information about the

Copyright © 1995-2020 Esri. All rights reserved.

types of operational layers please refer to the topic Layer types. Typically, operational layers contain data that changes more frequently than that of a basemap.

To add an operational layer to the map, add the layer to the map's operational layers list.

```
// create a service feature table of wildfire response areas
var wildfireResponseUri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/
Wildfire/FeatureServer/2");
var wildfireAreasFeatureTable = new ServiceFeatureTable(wildfireResponseUri);

// create a new feature layer from the table
var wildfireAreasLayer = new FeatureLayer(wildfireAreasFeatureTable);

// add the layer to the OperationalLayers collection in the map
myMap.OperationalLayers.Add(wildfireAreasLayer);
```

A layer that is added first is placed directly above the basemap and is in the 0th position in the map's operational layers list. If you add additional layers they are placed above the last layer and their position will be incremented by one. So the next layer will be in the first, then the next in the second position, and so on.

Note:
If your code adds operational layers to an existing map, the map's original list of operational layers is overridden to contain only the added layers. To keep the map's original operational layers and add new ones, wait until the map is loaded before adding operational layers. You can declare new operational layers outside the map's declaration, and use a signal handler to wait for the map to finish loading before adding the new layers. See the next section of this topic for details about loading a map and its layers.

If you wish to, you can insert a layer into the list of operational layers at a specific position.

```
// insert a layer into a specified position of the OperationalLayers collection
//myMap.OperationalLayers.Insert(3, wildfireAreasLayer);
```

All operational layers are added to the map in the same way and also conform to the asynchronous loading pattern as detailed in the Loadable pattern topic.

Operational layer properties

Each operational layer has a set of default properties. You can modify these properties so that they are more appropriate for your map. For example, you can:

- Set the maximum and minimum scale at which the layer is visible.
- Change the opacity of the layer.
- Change the visibility of the layer.
- Override the rendering.
- Specify a definition expression.
- Change the order in which the layers are displayed in the map.

By adding operational layers to a group layer, you can control some of these properties on a group of layers simultaneously.

For more information please explore the specific layer and sublayer classes.

Copyright © 1995-2020 Esri. All rights reserved.

Load the map and its layers

The map has been designed to load as quickly as possible. You will be able to access the map and set its properties and iterate through its layers even if a layer has failed or a layer's service response is slow. One broken layer should not prevent the entire map from being used. A map should not load layers that are not needed. The advantage of this design is that the map can be used without having to wait for all of the layer's to be available.

When you construct the map, its load status will be `NotLoaded`. You can load the map (initiate the map's load cycle) either by:

- Calling `LoadAsync` on the map

```
// load the map
myNewMap.LoadAsync();
// check the load status and display an error message if necessary
if (myNewMap.LoadStatus == LoadStatus.FailedToLoad)
{
    StatusTextBlock.Text = "Unable to load map.";
}
```

- Or, by setting the map on the mapview.

```
MyMapView.Map = myNewMap;
```

Upon completion of the map's load cycle, the map's load status will be `Loaded`, indicating that:

- The map's spatial reference has been defined by the basemap's spatial reference. The Esri basemaps defined in the API all use Web Mercator.
- The map's initial viewpoint has been specified, for example, by setting the map's latitude, longitude and level of detail. (The `levelOfDetail` value sets the scale or resolution at which the map is displayed.)
- The map's list of layers has been created.

The map can be in a loaded state and ready for user interaction even if the individual layers have not finished loading. The map's layers could be in a `Loaded`, `NotLoaded`, or `FailedToLoad` state. Please explore the Loadable pattern for more information on loading maps and layers.

Save the map to a portal

You have designed and built a map. Now you can save the map so you, or other users, can view at a later time. To save the map to the ArcGIS Enterprise portal or your organization's portal the following conditions must be met:

- The map must be in a loaded state.
- Any layers that you wish to save must be in a loaded state.
- You must be authenticated to a portal.

Have the map and layers been loaded?

Before attempting to save the map it is prudent to confirm that the map is loaded and that the layers are loaded. If a layer has failed to load you may decide to save the map without the layer or investigate why you cannot access the layer. This will depend on your workflow.

Note:

If a layer has an invalid URL then the layer will have a layer type of Unknown Layer. This layer will not be saved to the map.

Copyright © 1995-2020 Esri. All rights reserved.

```
// check the map load status and display an error message if necessary
if (myNewMap.LoadStatus == LoadStatus.NotLoaded)
{
    StatusTextBlock.Text = "Map is not loaded.";
}

// see if the map layers loaded successfully
foreach(var lyr in myNewMap.AllLayers)
{
    // report errors for layers that didn't load
    if(lyr.LoadStatus == LoadStatus.FailedToLoad)
    {
        StatusTextBlock.Text = "Error loading layer " + lyr.Name +
            ": " + lyr.LoadError.Message;
    }
}
```

Has the user authenticated with the portal?

When your user authenticates to a portal (including ArcGIS Online) they have access to the maps that they own. When they log into ArcGIS Online or ArcGIS Enterprise portal, these maps are visible in a set of folders under the **My Content** tab. In ArcGIS Enterprise portal, the My Content filter is selected by default. When a user saves a map with this API, the map is saved into a folder under My Content and is accessible from there.

To authenticate a user to a portal you must create a portal object, provide the essential credentials and then asynchronously load the portal. Just like map and layer classes, the `ArcGISPortal` class conforms to the Loadable pattern.

```
// get the user credentials (username and password); add it to the authentication manager
var cred = await AuthenticationManager.Current.GenerateCredentialAsync(
    new Uri(PortalUriTextBox.Text),
    UserTextBox.Text,
    PasswordTextBox.Password);
AuthenticationManager.Current.AddCredential(cred);

// pass the credentials (token) and create the portal instance
var myPortal = await ArcGISPortal.CreateAsync(new Uri(PortalUriTextBox.Text), cred,
CancellationToken.None);
```

For more information on authenticating with a portal, see Access the ArcGIS Platform.

Save the map

Saving a map to the portal creates a new web map portal item. The map is placed in a folder of your choice in the My Content area of the authenticated user. For full details on how to save the map to a portal, see Save a map.

Share the map

By default, the map that the user adds to the portal is only accessible to that user. The map is not available to others. For example, the map does not appear in search results and is not part of any group. There are several options for sharing a map. The options are described in the Share a portal item topic.

Related topics

Maps and scenes

Copyright © 1995-2020 Esri. All rights reserved.

Display a map
Wraparound maps
Save a map

Copyright © 1995-2020 Esri. All rights reserved.

Wraparound maps

A globe provides a better representation of the earth than a flat map does. In addition to more accurately displaying the shape of the earth, a globe provides seamless interaction with the features it displays. When spinning a globe, for example, it does not stop when it reaches a specific location (longitude). While a flat map has to divide the earth along a line of longitude (usually at 180º east/west), there's no reason why a map displayed on a computer screen can't wraparound to provide a continuous display of features as you pan east or west.

Note:
You may hear several different terms to describe the longitude at 180º east/west. While the following terms are not completely analogous, they are often used to refer to this line of longitude: `Dateline`, `International Dateline`, and `Antimeridian`.

Enable or disable wraparound

By default, a map view attempts to wrap your map for a continuous experience as the user pans the map east and west. To disable wraparound behavior for a map view (or to re-enable it), you can set the wraparound mode to the appropriate value. Wraparound can only be applied to a map view if the following requirements are met.

- The map's full extent covers the world.
- The map's spatial reference is WGS 84 (WKID=4326) or Web Mercator (WKID=102113, 102100, or 3857). This means that all tiled layers in the map must belong to one of these spatial references. Dynamic layers, however, can be in any spatial reference because they are capable of reprojecting their data.
- Dynamic layers in the map are based on services from ArcGIS Server 10.0 or higher. Earlier versions of the REST API do not support well-known text (WKT) values for spatial reference, which is required for making dynamic map services support wraparound.

If these requirements are not met, attempting to enable wraparound will fail silently (without raising an error, in other words).

The following example disables wraparound behavior if it is enabled and attempts to enable wraparound if it is not enabled.

```
// toggle the wrap around setting for the map view
if (MyMapView.WrapAroundMode == WrapAroundMode.EnabledWhenSupported)
```

Copyright © 1995-2020 Esri. All rights reserved.

```
{
    // disable wrap around
    MyMapView.WrapAroundMode = WrapAroundMode.Disabled;
}
else
{
    // try to enable wrap around
    MyMapView.WrapAroundMode = WrapAroundMode.EnabledWhenSupported;
          // check if it is now enabled
    if (!MyMapView.IsWrapAroundEnabled)
    {
        // ... wrap around was not enabled because it is not supported
        MessageTextBlock.Text = "Unable to enable wrap around";
    }
}
```

Normalize geometries

To better understand wraparound, it is helpful to visualize a map as being composed of frames. The portion of a map visible when wraparound is not enabled is frame 0. Taking an example of a map with the WGS 84 coordinate system, frame 0 has X coordinates between -180° (west) and +180° (east) longitude.

Adjacent to this frame to the east is frame 1, which extends hypothetically between +180° and +540° longitude. If you have panned across the map eastwards until you have passed 180°, you will be viewing frame 1. Similarly, adjacent to frame 0 to the west is frame -1, which extends hypothetically between -180° and -540° longitude. The series of frames continues infinitely in both directions.

Longitude values (X coordinates) returned from a map may be **real**: in the range of -180° and +180° (within frame 0 in other words) or they may be **hypothetical**: less than -180° or greater than +180° (outside of frame 0). Here are some examples of geometries that may contain hypothetical coordinates if wraparound is enabled:

- Map extent, returned from `MapView.GetCurrentViewpoint()`.
- Map locations, returned from `MyMapView.ScreenToLocation()`.
- Geometries digitized by a sketch layer

The process of converting a geometry so that it contains only real coordinate values is called **normalization**. Geometries that contain hypothetical values are not acceptable as inputs to spatial queries, projection, geocoding services, or routing services or for storage in a geodatabase. Because it may be difficult to determine if a complex shape contains hypothetical coordinate values, it's usually more efficient to simply normalize all geometry from the map when wraparound is enabled.

You can normalize geometries using `NormalizeCentralMeridian` on the `GeometryEngine` class.

```
// get the current map extent
var extent = MyMapView.GetCurrentViewpoint(ViewpointType.BoundingGeometry).TargetGeometry;
// if wraparound is enabled, normalize the geometry (in case all or part is outside of frame 0)
if (MyMapView.IsWrapAroundEnabled)
```

Copyright © 1995-2020 Esri. All rights reserved.

```
{
    extent = GeometryEngine.NormalizeCentralMeridian(extent);
}
```

The normalized geometry of a map extent that crosses the 180th meridian is a polygon consisting of two rings (split at that meridian).

Related topics

Maps and scenes
Display a map
Build a new map
Save a map

Copyright © 1995-2020 Esri. All rights reserved.

Save a map

You can create a map by building a new map or by using an existing map, as described in Display a map. Once constructed, the map includes both properties, for example:

- Map title and description
- Copyright information
- A thumbnail image
- Ratings

and map content, for example:

- The map's basemap
- A list of operational layers
- Information about layer visibility and rendering
- Layer scale thresholds
- The map's initial viewpoints

No matter how you create this map you can save it either as a brand new map or, if you have made any changes to the map, you can save it back to its original map. This topic describes how to save a brand new map and how to save back any changes you've made to an existing map.

Save a new map to a portal

When you save a new map to the portal you will create a new portal item with a type of `WebMap` that has a new unique item ID. The new portal item will be placed in the **My Content** area of the user that was authenticated with the portal. When you save the map for the first time you must provide at least a map title, a set of tags used to describe the map for searches, and a loaded portal object. You can also set these properties if you wish.

- **Folder**—a portal folder `(PortalFolder)` in which the map will be placed in My Content. By default the map will be saved to the user's root folder.
- **Description**—a description of the map.
- **Thumbnail**—an image (`RuntimeImage`) that will be associated with the portal item and is shown on the portal item details page in ArcGIS Online or ArcGIS Enterprise.
- **Force save to supported version**—a boolean flag to indicate whether or not the map should be saved as the web map version supported by the API. This may cause data loss, as data unknown to the supported format will be lost when the map is saved.

The following example illustrates saving the map with the asynchronous `SaveAsAsync` method and returning a new `PortalItem`.

```
// export the current map view to use as the thumbnail
var thumbnailImage = await MyMapView.ExportImageAsync();

// save the map to create a new portal item
var newPortalItem = await MyMapView.Map.SaveAsAsync(
    myPortal, // a portal with which the user has authenticated
    null, // folder ID (or null for the user's default folder)
    TitleTextBox.Text, // title
    DescriptionTextBox.Text, // description
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    TagsTextBox.Text.Split(','), // tags (keywords) used to identify the item in a search
    thumbnailImage, // the exported RuntimeImage for the item's thumbnail
    true);  // force the save to use the supported web map version
```

Note:

The overload of `SaveAsAsync` that takes a `System.Uri` for the thumbnail image has been deprecated. If passing a thumbnail image, use the recommended overload that takes an `Esri.ArcGISRuntime.UI.RuntimeImage` object.

Access the map's item properties

After the map saving has completed, the map's portal item property is populated with the brand new portal item properties. You can access all of the properties and set any of the writeable properties, as you require.

```
// get the portal item that represents the map
PortalItem webMapPortalItem = MyMapView.Map.Item as PortalItem;
  // provide a new title for the portal item
webMapPortalItem.Title = TitleTextBox.Text;

// provide a 'Snippet' (a more succinct version of the description)
webMapPortalItem.Snippet = SnippetTextBox.Text;

// report how many times the item has been viewed
var message = "Web map portal item (id=" + webMapPortalItem.ItemId + ")" +
    " has been viewed " + webMapPortalItem.ViewCount.ToString() + " times.";
```

If you wish to save any of these properties, call the map's save method as described in the following section.

Share the map

After the map has been saved to the portal, the map is available to share with other portal users. See Share a portal item for more information.

Save changes to an existing web map

If you have opened a map, made changes to the map content or properties, and you want to save changes back to that map, then call the asynchronous save method on the map. This method, of course, requires that the map has been previously saved to the portal and an exception will be raised if it has not. If a layer in your map allows users to edit offline, be sure to sync their offline edits in addition to saving the map.

```
try
{
    // save changes to the portal item properties
    await MyMapView.Map.SaveAsync();
}
catch(Exception ex)
{
    Console.WriteLine("Unable to save the web map portal item: " + ex.Message);
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Related topics

Maps and scenes
Display a map
Build a new map
Wraparound maps

Copyright © 1995-2020 Esri. All rights reserved.

Add a layer

There are two common roles for layers:

- Basemap layers - provide context (a background) for your data
- Operational layers - the map data actively in use

For example, a map of a utility network might show pipes in an operational layer and a street map background (basemap) to make it clear which houses those pipes serve.

To learn more about layer concepts, see Layers and tables. To learn about supported layer types and the details for each, see Layer types described.

If you want to quickly visualize data that's available to your app, but don't want to store it in an ArcGIS portal or a layer, consider using graphics.

Operational layers

Maps and scenes expose an operational layers property, which is a collection of layers. You can add and remove layers from the map's operational layers.

```
Map map = new Map();
map.OperationalLayers.Add(someLayer);
map.OperationalLayers.Remove(someLayer);
```

Basemap layers

Basemaps exposes a 'base layers' property, which contains all of the layers in the basemap. The BaseLayers property can be accessed through the basemap property on maps and scenes. Note that using a basemap for all your maps is recommended, when a suitable one is available. Some examples of basemaps that can be created using static factory methods are:

- World imagery
- World streets
- National geographic
- World navigation (vector)
- World streets (vector)

```
// Use the navigation basemap static method
Map map = new Map(Basemap.CreateNavigationVector());
// or
map.Basemap = Basemap.CreateNavigationVector();
// or
map.Basemap.BaseLayers.Add(someBasemapLayer);
```

Adding a layer to muliple maps or scenes

A layer can be added only once to a single map or scene. If you add a layer already in one map or scene to another map or scene then an exception will be thrown with the message: `object is already owned` (see ArcGIS Runtime errorsTo use the same layer in more than one map or scene you have a few options:

Copyright © 1995-2020 Esri. All rights reserved.

- If you need the layer to be in two map or scene objects (or one of each!) you can add a copy of the layer to your second map or scene.

  Note:
  The copy/clone method takes deep copy of the specific layer type, rather than the `Layer` base class. Any changes made to the original layer will not be reflected in the copied version of the layer.

- Create a new layer from the original source data. This is not the recommended approach because it requires accessing the underlying data source twice. For example, in the case of layers made from an online source, it would mean accessing the associated service twice.
- If the layer is no longer required in the first map or scene, remove the layer from the first map or scene and add it to the second map or scene.

Copyright © 1995-2020 Esri. All rights reserved.

Work with scenes (3D)

Copyright © 1995-2020 Esri. All rights reserved.

Display a scene

Scenes allow you to display and share collections of 3D geographical data across the ArcGIS platform. This guide explains how to display a pre-configured scene that is stored within a web scene or within a mobile scene package. The difference between 2D maps and 3D scenes is described in the topic Maps and scenes.

Note:
If you want to learn how to create your own scene in your app using code, see Build a new scene.

Open a scene from a web scene

Web scenes can be accessed using either their URL or directly from a portal using their portal item ID. Both methods are described below along with the authentication process required to access the scene if it, or its data is not public.

Using a web scene URL

You can open a scene directly from a URL in one of the following formats:

- **Scene viewer URL** - `https://www.arcgis.com/home/webscene/viewer.html?webscene=31874da8a16d45bfbc1273422f772270`
- **Item URL** - `https://www.arcgis.com/home/item.html?id=31874da8a16d45bfbc1273422f772270`
- **Item data URL** - `https://www.arcgis.com/sharing/rest/content/items/31874da8a16d45bfbc1273422f772270/data`

```
// Create the scene from the URL.
Scene sceneBerlin = new Scene(new Uri("https://www.arcgis.com/home/
item.html?id=31874da8a16d45bfbc1273422f772270"));
```

If the scene is not public, the API's authentication challenge handler automatically requests that the user provides their credentials.

Using a web scene portal item

You can construct a scene object using the scene's portal item ID. You can find this 32 character ID within any of the three URLs described above.

```
// Initialize the portal with ArcGIS Online.
ArcGISPortal portalAGOL = await ArcGISPortal.CreateAsync();

// Get the portal item.
PortalItem portalItemBerlin = await PortalItem.CreateAsync(portalAGOL,
"31874da8a16d45bfbc1273422f772270");

// Create the scene from the portal item.
Scene berlinScene = new Scene(portalItemBerlin);
```

Copyright © 1995-2020 Esri. All rights reserved.

Authentication

If the scene or its layers are not public, you need to provide credentials to access them. In these cases you can either: rely on the API's authentication challenge handler to prompt the user for their credentials, or you can provide your own login facility and pass the credentials to the portal's credential property as described in the Access the ArcGIS platform topic.

Open a scene from a mobile scene package

Mobile scene packages allow you to work with scenes, 3D layers, elevation sources and associated data in a disconnected environment. You can create a mobile scene package (`.mspk`) file using ArcGIS Pro and deliver it using a platform-specific transfer mechanism such as AirDrop, or download it from a portal to the device. For more information, see Create mobile scene package.

To access the scenes in a mobile scene package:

1. Construct the mobile scene package object from the (`.mspk`) file path, or an unpack directory, if necessary.

```
// Get the path to the mobile scene package.
string filepath = GetPackagePath();

// Open the scene package.
MobileScenePackage scenePackage = new MobileScenePackage(filepath);
```

2. Load the mobile scene package to access or display any of the scenes in the package. Each package can contain one or more scenes.

```
try
{
    // Open the scene package.
    await scenePackage.LoadAsync();

    // Check that there is at least one scene.
    if (scenePackage.Scenes.Count > 0)
    {
        // Display the first scene in the package.
        MySceneView.Scene = scenePackage.Scenes.First();
    }
}
catch (Exception e)
{
    Console.WriteLine("Error:" + e.ToString());
}
```

If you want to obtain or set any of the scene's properties before displaying it, then load the scene first using the loadable pattern. If not, then just pass the scene to the scene view and the scene will be loaded automatically.

Display a scene in a scene view

To display the scene, pass the `Scene` object to the `SceneView`. This provides access to the scene, its layers and data, sets the camera's viewpoint on the scene, and enables a range of 3D navigation tools for your app users.

```
MySceneView.Scene = sceneBerlin;
```

Copyright © 1995-2020 Esri. All rights reserved.

Monitor scene drawing

The scene is redrawn whenever the user:

- pans, rotates or zooms
- a camera controller updates the camera
- data is added or removed from the scene

You can subscribe to the `SceneView.DrawStatusChanged` event to be notified whenever the drawing status changes.

```
// Handle the DrawStatusChanged event for the scene view.
MySceneView.DrawStatusChanged += (s, info) =>
{
    Console.WriteLine($"SceneView drawing status: {info.Status}");
};
```

Navigate the scene

The scene defines the initial viewpoint. Users can then navigate the scene using the mouse, keyboard or touch screen, as available. See Navigate a scene view to learn about the various interaction options and how to programmatically navigate the scene.

Configure the scene view

There are a number of important concepts to consider when configuring a scene view:

- Camera controllers - define how navigation operates in the scene. For example, they can be used to follow a feature or graphic, or to orbit a specific location. To learn about camera controllers, see Follow a graphic in a scene view.
- Lighting - you can set the ambient light color (`ambientLightColor`), specify whether there are shadows (`sunLighting`) and control the position of the sun by setting the date and time on the `sunTime` property
- Atmosphere effect - defines how the sky looks. There are three observer effect options:
  - **Realistic** - There is an atmosphere effect applied to both the sky and the surface as viewed from above.
  - **Horizon only** - There is an atmosphere effect applied to the sky (horizon) only.
  - **None** - There is no atmosphere effect; anything that isn't a part of the scene is rendered black with a starfield consisting of randomly placed white dots.

  You can apply any of these options to the scene view's `AtmosphereEffect` property.

Copyright © 1995-2020 Esri. All rights reserved.

The background rendered by the `SceneView` is configured by the `SpaceEffect` property. The space effect defaults to `Stars`, which will render a black background with a field of stars. You can turn off background rendering by setting the space effect to `None`. The `None` space effect is useful for augmented reality experiences where you display a camera feed in the background.

Filter data by time

Some layer types contain data that can be filtered by time values. Runtime has rich support for time-aware layers and data. See Visualize and compare data over time to learn more.

Copyright © 1995-2020 Esri. All rights reserved.

Build a new scene

Scenes are the fundamental document type used for persistence and sharing of 3D content across the ArcGIS platform. To learn about opening scenes from ArcGIS Online and Portal, see Display a scene. This guide describes how to create a new scene programmatically, add content, and display the scene in your app.

The data you visualize in your 3D app is defined by the scene class. The types of data you can display in a scene include:

- Basemaps, which can be draped over surface layers. A surface layer contains elevation information. Other layer types can also be used as a basemap; there are no special restrictions for basemaps in scenes.
- Operational layers, which can include feature layers or scene layers. Scene layers represent built and natural 3D content such as buildings, trees, valleys, and mountains and can be accessed as scene services or local scene layer packages.
- Surface layers, which define the elevation information of the 3D visualization. A surface layer can come in many forms, for example, local raster data such as a DEM or DTED or from an ArcGIS image service.

To learn more about the types of layers that can be used in scenes, see Layer types described. To learn more about the content that can be used in maps and scenes, see Layers and tables.

Add a scene layer from a service or package

1. Create a scene that contains a basemap and a scene layer, either from a service or a local package.

```
// create a new scene with a topographic basemap
var myScene = new Scene(Basemap.CreateTopographic());

// create a scene layer to display (3D) buildings
var dddBuildingsLayer = new ArcGISSceneLayer(new System.Uri("http://scene.arcgis.com/arcgis/
rest/services/Hosted/Buildings_Brest/SceneServer/layers/0"));

// add the scene layer to the scene
myScene.OperationalLayers.Add(dddBuildingsLayer);
```

2. To display the scene, assign it to a scene view that you create.

```
// add a scene to the scene view
MySceneView.Scene = myScene;
```

3. Run the project and it should resemble the following image. You can use your mouse to pan and zoom around the scene. For more information on navigating the scene, see the Navigate a scene view topic.

   You may want to set the camera (set an initial viewing position) or add graphics.

Copyright © 1995-2020 Esri. All rights reserved.

Add a 3D surface (terrain)

Add a surface layer to a scene with a basemap for 3D visualization in your app.

A basemap added to a scene is flat if no surface data has been added

The same basemap added to a scene is draped onto the terrain if surface data has been added

1. Create a scene and assign it to a scene view.

Copyright © 1995-2020 Esri. All rights reserved.

```
// create a new scene with a topographic basemap
var myScene = new Scene(Basemap.CreateTopographic());

// add the new scene to the scene view
MySceneView.Scene = myScene;
```

2. Add an elevation surface and apply it to the scene.

```
// create an elevation source
var elevationSource = new ArcGISTiledElevationSource(new
System.Uri("http://elevation3d.arcgis.com/arcgis/rest/services/WorldElevation3D/Terrain3D/
ImageServer"));
// create a surface and add the elevation surface
var sceneSurface = new Surface();
sceneSurface.ElevationSources.Add(elevationSource);

// apply the surface to the scene
MySceneView.Scene.BaseSurface = sceneSurface;
```

You can use your mouse to pan and zoom around the scene. For more information on navigating the scene, see the Built-in navigation section of the "Navigate a scene view" topic.

Subsurface navigation

The camera can be navigated below the surface when the surface has been enabled to allow subsurface navigation. To allow the camera to move below the surface set the surface's navigation constraint to `None`. If you wish to keep the camera above the surface set the surface's navigation constraint to `Stay Above`, the default value.

The opacity property of the surface object allows users to see objects both above and below the surface. Opacity has a multiplicative effect on the opacity of each of the base layers and the background grid.

Set the camera

The position you view the scene from is defined by a Camera. Define the following properties when you create a new camera:

- 3D location—Latitude, longitude, and altitude
- Heading—Direction the camera is pointing
- Pitch—Up and down angle of camera

For example, to point the camera to toward the Snowdon mountainside, as shown in the following code snippet and image, use these values:

- For 3D location, use 53.06 latitude, -4.04 longitude, 1289 metres above sea level
- For heading, use 295 degrees
- For pitch, use 71 degrees

```
var snowdonCamera = new Camera(53.06, -4.04, 1289, 295, 71, 0);
```

Copyright © 1995-2020 Esri. All rights reserved.

Now you have a new camera you can apply to your scene view. You can apply it immediately using `SetViewpointCamera` as shown in the code below, or the camera can be animated to the new position using one of the asynchronous methods.

```
MySceneView.SetViewpointCamera(snowdonCamera);
```

Create a scene layer package and local elevation source

ArcGIS Pro allows you to create a scene layer package using a geoprocessing tool. You can add this local scene layer package (`.slpk`) as a scene layer (`ArcGISSceneLayer`) to your ArcGIS Runtime app and add it to a `Scene`. To display the scene layer package, create a new scene layer with a URL pointing to a local scene layer package.

To take this scene layer offline your app can define an elevation surface by consuming local raster datasets on the client. The following formats are supported for creating a local elevation source:

- ASRP/USRP
- CIB1, 5, 10
- DTED0, 1, 2
- GeoTIFF
- HFA
- HRE
- IMG
- JPEG
- JPEG 2000
- NITF
- PNG
- RPF
- SRTM1, 2

Related topics

Navigate a scene view
Add features and graphics to a scene view

Copyright © 1995-2020 Esri. All rights reserved.

Navigate a scene view

You can navigate a scene view (a 3D map):

- Using built-in navigation (described on this page), such as panning, zooming, and changing pitch
- By programmatically changing camera position

Built-in navigation

The following table summarizes the built-in navigation capabilities of the `SceneView` control.

Navigation	Mouse	Touch
Pan	• Hold left button and drag	• Drag • Flick
Zoom in	• Scroll wheel forward • Hold center button and drag	• Spread
Zoom out	• Scroll wheel backward • Hold center button and drag	• Pinch
Rotate	• Hold right button and drag	• Multiple finger hold and twist
Change pitch		• Multiple fingers drag up/down

Copyright © 1995-2020 Esri. All rights reserved.

Navigation	Mouse	Touch
Change heading		• Multiple fingers drag left/right

Navigate with touch

The `SceneView` control provides built-in support for navigating a scene using touch.

There are three navigation modes available using touch: single finger pan, multiple finger zoom and pan, and multiple finger rotate and pitch. Place one finger on the display to use single finger pan. Spread, pinch, or drag two or more fingers on the display to enter pan and zoom mode, or place multiple fingers on the display at a consistent distance to navigate in heading and pitch mode. The following list describes the touch gestures used to execute navigation operations in each mode.

- Pan mode (single finger)
  - Drag—Pans
  - Flick—Pans
- Pan and zoom mode (multiple fingers)
  - Spread—Zooms in
  - Pinch—Zooms out
  - Drag—Pans
  - Pivot—Rotates
- Heading and pitch mode (multiple fingers)
  - Drag up and down—Changes pitch
  - Drag left and right—Changes heading

Tip:

If the desired navigation gesture is not initially recognized, remove your fingers and replace them on the display.

Programmatically change camera position

Your applications can programmatically navigate a 3D scene by creating a new camera and setting it to the scene view. A camera defines the location from which you are viewing the scene.

Copyright © 1995-2020 Esri. All rights reserved.

The camera is shown in this image for illustration purposes; when you set camera settings (location, pitch), think of the camera class a real-life camera you're adjusting the position of.

You can enable subsurface navigation in code by

Set the camera

For example, to point the camera to toward the Snowdon mountainside, as shown in the following code snippet and image, use these values:

- For 3D location, use 53.06 latitude, -4.04 longitude, 1289 metres above sea level
- For heading, use 295 degrees
- For pitch, use 71 degrees

```
var snowdonCamera = new Camera(53.06, -4.04, 1289, 295, 71, 0);
```

Now you have a new camera you can apply to your scene view. You can apply it immediately using `SetViewpointCamera` as shown in the code below, or the camera can be animated to the new position using one of the asynchronous methods.

```
MySceneView.SetViewpointCamera(snowdonCamera);
```

Copyright © 1995-2020 Esri. All rights reserved.

Related topics

Build a new scene
Add features and graphics to a scene view

Copyright © 1995-2020 Esri. All rights reserved.

Add features and graphics to a scene view

In ArcGIS Runtime SDKs, **geoelements** are used to represent real-world objects on a map or scene. The two types of geoelement are **features** and **graphics**, each of which has a geometry representing its shape and location on the surface of the Earth. They can also contain a set of attributes that further describe the represented object.

You can add graphics to graphics overlays and add graphics overlays to either a map view or scene view. The way you add features and graphics to scenes follows the same steps required to add them to maps, with these additional considerations:

- Surface placement modes
- 3D rendering properties
- 3D-specific symbols

When to use features or graphics

Features and graphics are similar to one another in that they both have a geometry and may store a set of attributes. Both features and graphics can be:

- Displayed at a location above or below sea level or the surface layer (If the data includes z-values). See Surface placement modes.
- Extruded according to a chosen extrusion mode and expression. See 3D rendering properties.
- Rendered either statically or dynamically. See Rendering modes.

Features and graphics, however, are designed for different uses. While both can represent geographic elements, there are factors that may make one of them a better choice for specific tasks. See Features and graphics for more information.

Add features to a scene view when you want to:

- Display data with a common attribute schema (for example, census data)
- Display geographical data of a common geometry type (point, line, or polygon)
- Persist or store the data beyond the current app session

Add graphics to a scene view when:

- You want to display temporary data, such as fast-changing, fast-moving data (for example, a fleet of airliners, or voluminous data).
- You want to display data that doesn't share a common geometry type (such as point, line, or polygon) and doesn't share a common set of attribute fields
- The data is required only for the current app session

Add features to a scene view

Feature layers can contain any number of features of a single geometry type (point, polyline or polygon). Adding features to a scenes is the same as adding features to a map:

1. Add features to a feature layer
2. If you want to include features' attributes (to run queries, for example), make sure to use `QueryFeatureFields.LoadAll` if building your feature layer from a service feature table
3. Add the feature layer to the scene's operational layers

Copyright © 1995-2020 Esri. All rights reserved.

Add graphics to a scene view

Adding graphics to scenes is the same as adding graphics to maps:

1. Add graphics to a graphics overlay
2. Set the graphic overlays surface placement, if you prefer a surface placement other than the default `SurfacePlacement.Draped`
3. Add the graphics overlay to the scene view's graphic overlays

Note:
Any graphic you add to a scene view must be constructed from a geometry with a spatial reference.

In 3D, you can add graphics such as points, polylines, polygons, and 3D marker symbols, all into the same graphics overlay or into multiple graphics overlays.

For additional details on adding graphic overlays to a map or scene, see Add graphic overlays to your app. For details on adding graphics to a graphics overlay, see Add graphics and text to graphics overlays.

```
var sceneGraphicsOverlay = new GraphicsOverlay();
// ... set graphics overlay properties ...
MySceneView.GraphicsOverlays.Add(sceneGraphicsOverlay);
```

Surface placement modes

For both feature layers and graphics overlays, you can specify a surface placement mode. These modes control how the z-value of your geometry is used when features and graphics are drawn.

- `DrapedBillboarded`—Ignore Z values and drape symbols onto the surface, billboarded to always face the camera.
- `DrapedFlat`—Ignore the Z values and drape on the surface.
- `Absolute`—Features and graphics are drawn at a height using the z-value referenced from above the globe skin (sea level).
- `Relative`—Features and graphics are drawn at a height using the z-value referenced from above the surface layer.
- `RelativeToScene`—Placement is relative to objects in the scene, such as an elevation surface, buildings in a scene layer, integrated mesh layers, and extruded polygons. Some objects do not affect relative to scene placement, such as extruded points or lines, KML layers, or model symbols. Some combinations may not work for relative to scene placement, such as polygons placed relative to an integrated mesh layer.
- `Draped`—(obsolete) Use `DrapedBillboarded` or `DrapedFlat` instead.

Note:
The minimum z-value is `-6,356,752` meters, which is the approximate radius of the earth (the WGS 84 datum semi-minor axis). The maximum z-value is `55,000,000` meters. ArcGIS Runtime may not correctly display geometry that has z-values outside of this range.

The following code shows three points drawn in graphics overlays in the three different surface placement modes. The red graphic is draped on the surface, the blue graphic is drawn in absolute mode, and the green in relative mode.

Copyright © 1995-2020 Esri. All rights reserved.

```
// create and add a new graphics overlay with Draped surface placement
var drapedOverlay = new GraphicsOverlay();
drapedOverlay.SceneProperties.SurfacePlacement = SurfacePlacement.DrapedBillboarded;
MySceneView.GraphicsOverlays.Add(drapedOverlay);
// create and add a new graphics overlay with Absolute surface placement
var absoluteOverlay = new GraphicsOverlay();
absoluteOverlay.SceneProperties.SurfacePlacement = SurfacePlacement.Absolute;
MySceneView.GraphicsOverlays.Add(absoluteOverlay);
// create and add a new graphics overlay with Relative surface placement
var relativeOverlay = new GraphicsOverlay();
relativeOverlay.SceneProperties.SurfacePlacement = SurfacePlacement.Relative;
MySceneView.GraphicsOverlays.Add(relativeOverlay);

// create three marker symbols (red, green, and blue circles)
var redCircleSym = new SimpleMarkerSymbol(SimpleMarkerSymbolStyle.Circle, Color.Red, 10);
var greenCircleSym = new SimpleMarkerSymbol(SimpleMarkerSymbolStyle.Circle, Color.Green, 10);
var blueCircleSym = new SimpleMarkerSymbol(SimpleMarkerSymbolStyle.Circle, Color.Blue, 10);

// create a map point to display
var location = new MapPoint(-4.04, 53.06, 1000, SpatialReferences.Wgs84);
// create three graphics to display the location with different symbols
var redGraphic = new Graphic(location, redCircleSym);
var greenGraphic = new Graphic(location, greenCircleSym);
var blueGraphic = new Graphic(location, blueCircleSym);
// add each graphic to a different graphics overlay (different surface placement)
drapedOverlay.Graphics.Add(redGraphic);
absoluteOverlay.Graphics.Add(blueGraphic);
relativeOverlay.Graphics.Add(greenGraphic);
```

Graphics rendered at different heights, despite all having the same z-value.

Copyright © 1995-2020 Esri. All rights reserved.

3D rendering properties

When using renderers in 3D, you can set properties to define extrusion expressions for both features and graphics. For graphics you can also define a rotation expression. For more information and examples of using extrusion expressions with a renderer, see 3D renderer properties in the Symbolize data topic.

Extrusion is the process of stretching a flat 2D shape vertically to create a 3D object. This provides a simple method to create three-dimensional symbology from two-dimensional features. For example, you can extrude building polygons by a height value to create realistic building shapes. You can also use extrusion to illustrate relative values in your data. The three basic geometry types-points, lines, and polygons-all support extrusion

Note:
Extrusion may not work with some types of multilayer symbols, such as hatch fills.

The following image shows a graphics overlay of US counties extruded based on a population attribute.

Create 3D symbols

While you can use the same 2D symbols available for symbolizing graphics in a map view, ArcGIS Runtime provides some specialized symbols for displaying graphics in a scene view. You can display points and lines as 3D shapes or use models to realistically represent real-world objects (such as vehicles).

ArcGIS Runtime provides the following types of 3D-specific symbols. See 3D-specific symbols for more information.

- Simple marker scene symbols—Cone, Cube, Cylinder, Diamond, Sphere, and Tetrahedron
- Stroke symbol layer 3D line styles—Strip, Tube
- Model 3D symbols—Displays a 3D model file

Copyright © 1995-2020 Esri. All rights reserved.

Shape-based 3D symbols

Model 3D symbols

Related topics

Build a new scene
Navigate a scene view
Add graphics overlays to your app
Add graphics and text to graphics overlays
Features and graphics
Symbols, renderers, and styles

Copyright © 1995-2020 Esri. All rights reserved.

Follow a graphic in a scene view

Graphics can represent dynamic objects in the scene. A service might provide near real-time locations for a fleet of vehicles, for example. Your device might read GPS information and display your current location as a graphic, or you might simply animate graphics on the scene view by reading a hard-coded list of geographic coordinates. For these use cases, there are two ways in which your user might want to visualize moving graphics in a scene view. They might want to view several graphics at once as they interactively explore the entire scene, or they might want to continuously point the camera at a specific element and track it as it moves through the scene. Your app might even allow the user to interact both ways, sometimes selecting a particular graphic to follow, and then switching to explore the entire scene.

Visualize dynamic graphics in a scene

A scene view uses the concept of a camera and a focal point to display data in three dimensions. The camera represents the location of the observer in three-dimensional space (x, y, z), and the camera's focal point is the location that intersects the camera's line of sight. By default, a scene view allows the user to interactively change the camera position to center on a focal point of their choice. However, you may need to set the camera's focal point relative to a specific graphic or location in the scene view. In the case of a moving graphics, you may want to continuously update the camera to maintain focus on a particular graphic as it moves through the scene.

To simplify the code required to keep the camera in sync with a moving graphic, ArcGIS Runtime provides a `CameraController` to implement this behavior.

Note:
A camera controller does not honor obstacles (such as terrain, extruded buildings, or other graphics) as it follows a graphic. In other words, the camera does not move to avoid such obstacles as it follows a graphic.

Prepare your graphics

To ensure your graphics can be tracked and properly displayed using the method described in this topic, follow these guidelines:

- Graphics you want to follow should have attributes to store the current heading, pitch, and roll.
- Each graphics overlay containing graphics you want to follow should have a renderer assigned.
- The graphics overlay renderer should have heading, pitch, and roll expressions defined in the renderer scene properties. These expressions should use the corresponding graphic attributes.
- The graphics overlay that contains the graphic to follow must be added to the scene view.

Note:
Renderer scene property expressions can use hard-coded values, but they usually contain the name of an attribute. For example, the expression `[heading] + 3.5` indicates that the renderer will add 3.5 to the value of the attribute named `heading` for each graphic in the graphics overlay. When referring to attributes in an expression, the attribute name should be in square brackets, such as `[heading]`.

You may need to adjust the heading, pitch, and roll properties on your symbol to make sure its initial appearance is correct.

The following example creates a graphics overlay with the appropriate scene properties, adds a single point graphic to it, and then adds it to the scene view. The graphic `_planeGraphic` is now ready to follow in the scene.

Copyright © 1995-2020 Esri. All rights reserved.

```
// create a new RendererSceneProperties
var sceneProps = new RendererSceneProperties();

// set heading, pitch, and roll expressions that use the corresponding graphic attribute names
sceneProps.HeadingExpression = "[Heading]";
sceneProps.PitchExpression = "[Pitch]";
sceneProps.RollExpression = "[Roll]";

// create a new simple renderer and apply the scene properties
var graphicsRenderer = new SimpleRenderer();
graphicsRenderer.SceneProperties = sceneProps;

// create a new graphics overlay, define absolute placement of graphics, and apply the renderer
var followGraphicsOverlay = new GraphicsOverlay();
followGraphicsOverlay.SceneProperties.SurfacePlacement = SurfacePlacement.Absolute;
followGraphicsOverlay.Renderer = graphicsRenderer;

// create a point graphic to track
var startPoint = new MapPoint(-117.00, 31.50, SpatialReferences.Wgs84);
_planeGraphic = new Graphic(startPoint, planeSymbol);
_planeGraphic.Attributes["Heading"] = 270.0;
_planeGraphic.Attributes["Pitch"] = 45.0;
_planeGraphic.Attributes["Roll"] = 12.0;

// add the graphic to the overlay
followGraphicsOverlay.Graphics.Add(_planeGraphic);

// add the graphics overlay to the scene view
MySceneView.GraphicsOverlays.Add(followGraphicsOverlay);
```

Update graphic position

When using a camera controller to follow a graphic, update the position of the graphic and the camera updates its location accordingly. Provide a new `MapPoint` for the graphic's geometry, and update the heading, pitch, and roll attributes as needed. The graphics overlay renderer (and associated scene properties) controls updating the display of the graphic's heading, pitch, and roll.

This code updates the geometry and attributes of the graphic being tracked.

```
_planeGraphic.Geometry = newMapPoint;
_planeGraphic.Attributes["Heading"] = newHeading;
_planeGraphic.Attributes["Pitch"] = newPitch;
_planeGraphic.Attributes["Roll"] = newRoll;
```

Camera controllers

A scene view has an associated controller that manages the camera for the scene. Each type of camera controller is designed to provide a specific user experience for interacting with the scene display. The camera controller and its properties can be changed at run time, so your app can provide the scene interaction experience best suited for the current context.

ArcGIS Runtime SDK provides the following camera controllers:

- `GlobeCameraController` (default) —Provides the default scene view camera behavior. Allows the user to freely move and focus the camera anywhere in the scene.
- `OrbitGeoElementCameraController`—Locks the scene view's camera to maintain focus relative to a (possibly moving) graphic. The camera can only move relative to the target graphic.

Copyright © 1995-2020 Esri. All rights reserved.

- `OrbitLocationCameraController`—Locks the scene view's camera to orbit a fixed location (map point). The camera can only move relative to the target map point.

When any camera controller other than `GlobeCameraController` is active, the scene view's viewpoint cannot be assigned. Attempts to do so do not raise an exception, but they are ignored.

Note:
At this release, `OrbitGeoElementCameraController` works only with graphics.

Use a camera controller to follow a graphic

The `OrbitGeoElementCameraController` updates the camera focal point relative to a specified (stationary or moving) graphic. As the user interacts with the scene view, the scene view's camera follows a focal point specified by the target graphic. Using this controller, the camera moves in sync with the target.

Note:
By default, the target graphic is also the camera focal point. The focal point can also be placed relative to the target graphic by applying offsets. See Adjust the display for details.

The following example creates a new `OrbitGeoElementCameraController` and assigns it to the `SceneView`. The `Graphic` to follow is passed to the object's constructor, along with the initial distance from the target to the camera. By default, the camera is positioned with a heading of 0 and a pitch of 45 degrees. See Adjust the display for information about modifying camera settings.

```
// create an OrbitGeoElementCameraController, pass in the target graphic and initial camera
distance
var orbitGraphicController = new OrbitGeoElementCameraController(_planeGraphic, 1000);
MySceneView.CameraController = orbitGraphicController;
```

The graphic associated with an `OrbitGeoElementCameraController` cannot be changed after the controller is initialized. To follow another graphic, you must create a new `OrbitGeoElementCameraController` for the other graphic and set it as the scene view's camera controller.

Adjust the display

Although the `OrbitGeoElementCameraController` locks the camera to a target graphic, the exact location of the camera in relation to the target can be adjusted using offset values. Other aspects of the scene display can also be modified, such as whether the camera should automatically adjust its heading, pitch, and roll when those values change in the tracked graphic.

Copyright © 1995-2020 Esri. All rights reserved.

Adjustments to the camera display, as well as limiting user interaction with the scene view, can be changed dynamically at run time.

Camera target offsets

The position of the camera focal point relative to the target graphic can be adjusted using x, y, and z target offsets. This allows you to make precise adjustments to the display of the scene, for example, placing the camera inside the cockpit of an airplane graphic. These offsets can be assigned individually or applied at once using an animation.

The following example applies offsets for the `OrbitGeoElementCameraController`.

```
// set the x, y, and z camera target offsets (animate change over 2 seconds)
await orbitGraphicController.SetTargetOffsetsAsync(43.0, 0.0, -55.0, 2.0);
```

You can also use `TargetVerticalScreenFactor` to adjust the vertical position of the target in the display. Acceptable values range from 0 (bottom of the display) to 1 (top of the display). The default is 0.5, which places the target in the center of the display.

Auto heading, pitch, and roll

Auto heading, when enabled for the camera controller, means that the camera will rotate to follow changes in the graphic's heading and therefore provide a consistent view of the graphic. When auto heading is disabled, the camera remains at a constant heading and the graphic may appear to rotate as its heading changes.

Auto pitch and auto roll have similar effects. When enabled, the camera rotates with the target's pitch or roll and the surface appears to rotate around the graphic, as shown on the left in the following image. When disabled, the camera is not affected by changes in the graphic's pitch or roll.

These settings are `true` by default.

- `IsAutoHeadingEnabled`

Copyright © 1995-2020 Esri. All rights reserved.

- `IsAutoPitchEnabled`
- `IsAutoRollEnabled`

Move the camera

While the `OrbitGeoElementCameraController` is associated with the scene view, the camera can be moved relative to the target graphic (either programmatically or interactively).

- Camera distance—The distance in meters between the camera and its target. The camera position is derived from this distance plus the camera offsets.
- Camera heading offset—The clockwise angle measured in the target element symbol's horizontal plane starting directly behind the symbol (default is 0). Any angle in degrees is valid but will be normalized between the minimum and maximum camera heading offsets. If after normalization the value is outside of the allowed range, the nearest value within the range will be used.
- Camera pitch offset—The counterclockwise angle from the positive z-axis of the target element's symbol relative to the symbol's horizontal plane, as illustrated in the following image (default is 45). If the value specified is outside of the range defined by the minimum and maximum camera pitch offsets, the nearest value within the range will be used.

The following example moves the camera using the `OrbitGeoElementCameraController`.

```
// update camera position: delta values for distance, heading, and pitch (animate change over 2
seconds)
await orbitGraphicController.MoveCameraAsync(300.0, 0.0, 45.0, 2.0);
```

Note:
You can also change the offset properties individually as an alternative to using the `moveCameraAsync` method.

Limit user interaction

When using an `OrbitGeoElementCameraController` or `OrbitLocationCameraController`, you can define minimum and maximum values for the camera distance, heading, pitch, and roll properties if you want to further restrict user interaction with the scene view. User interactions and scene view animations obey these limits.

Tip:
Since the camera controllers don't recognize obstacles (such as terrain) when tracking a graphic, limiting or restricting the possible camera positions can help avoid them.

Copyright © 1995-2020 Esri. All rights reserved.

The following example limits the distance and pitch for the camera, but not the heading.

```
// limit the camera distance to between 1,000 and 10,000 meters
orbitLocationController.MinCameraDistance = 1000;
orbitLocationController.MaxCameraDistance = 10000;

// limit the camera pitch to between 25° and 75°
orbitLocationController.MinCameraPitchOffset = 25;
orbitLocationController.MaxCameraPitchOffset = 75;
```

Types of scene view interaction can also be completely disabled if needed. You may want the user to interact with the scene view by changing the camera heading and pitch, but not the camera distance, for example.

The following code disables the ability for the user to modify the camera distance when interacting with the scene view.

```
// restrict the ability to interactively change the camera distance
orbitLocationController.IsCameraDistanceInteractive = false;
```

Stop following a graphic

To stop following a graphic, reset the scene view's camera controller to the default, `GlobeCameraController`.

In general, the only time you need to assign this camera controller to the scene view is to return to the default behavior after applying one of the non-default controllers.

The following code applies a new `GlobeCameraController` to the `SceneView`.

```
// apply the (default) globe camera controller
MySceneView.CameraController = new GlobeCameraController();
```

The globe camera controller allows the user to interactively change the camera position, focal point, and any of the camera properties. Use this controller to let your user freely explore the scene. See Navigate a scene view for more information about interacting with a scene view using the globe camera controller.

Copyright © 1995-2020 Esri. All rights reserved.

Display scenes in augmented reality

ArcGIS Runtime provides support augmented reality through the AR Toolkit.

Note:
AR Toolkit is open source, so you can use it as is or modify its components to meet your needs. You can keep your changes proprietary or share them with the open source community.

AR experiences can be thought of in terms of three common patterns: tabletop, flyover, and world-scale. Runtime supports all three, and the rest of this document describes how to implement each.

- **Flyover** – Flyover AR is a kind of AR scenario that allows you to explore a scene using your device as a window into the virtual world. A typical flyover AR scenario will start with the scene's virtual camera positioned over an area of interest. You can walk around and reorient the device to focus on specific content in the scene.
- **Tabletop** – A kind of AR scenario where scene content is anchored to a physical surface, as if it were a 3D-printed model. You can walk around the tabletop and view the scene from different angles.
- **World-scale** – A kind of AR scenario where scene content is rendered exactly where it would be in the physical world. This is used in scenarios ranging from viewing hidden infrastructure to displaying waypoints for navigation. In AR, the real world, rather than a basemap, provides the context for your GIS data.

Flyover

On screen, flyover is visually indistinguishable from normal scene rendering.

Tabletop

In tabletop, scene content is anchored to a real-world surface.

World-scale

In world-scale AR, scene content is integrated with the real world.

Enable your app for AR using AR Toolkit

1. See the toolkit repo on GitHub for the latest instructions for installing: Toolkit repo on GitHubIf you're developing on Windows, consider installing the AR templates from Visual Studio Marketplace (Windows only). These templates simplify development of AR apps by providing pre-configured projects that are fully configured for AR.
2. Add an ARSceneView to your app.
3. Configure privacy and permissions.
4. Now you're ready to add tabletop AR , add flyover AR , or add world-scale AR to your app.

Copyright © 1995-2020 Esri. All rights reserved.

Add an AR view to your app

`ARSceneView` uses an underlying ARKit or ARCore view and an ArcGIS Runtime `SceneView` . `ARSceneView` subclasses `SceneView` , so you can access scene view properties, including the `Scene` property, directly. Use the following properties on `ARSceneView` to configure AR:

- `TranslationFactor` - controls the relationship between physical device position changes and changes in the position of the scene view's camera. This is useful for tabletop and flyover AR.
- `OriginCamera` - controls the initial position of the scene view's camera. When position tracking is started, `ARSceneView` transforms the scene view camera's position using a transformation matrix provided by ARKit or ARCore. Once the origin camera is set, the manipulation of the scene view's camera is handled automatically.
- `SetInitialTransformation` – takes a point on the screen, finds the surface represented by that point, and applies a transformation such that the origin camera is pinned to the location represented by that point. This is useful for pinning content to a surface, which is needed for tabletop AR.

In addition to AR Toolkit features, you'll need to use the following ArcGIS Runtime features provided by the underlying scene view when creating AR experiences:

- Scene view space effect control — Disable rendering the 'starry sky' effect to display scene content on top of a camera feed.
- Scene view atmosphere effect control — Disable rendering the atmosphere effect to avoid obscuring rendered content.
- Surface transparency — Hide the ground when rendering world-scale AR because the camera feed, not the basemap, is providing context for your GIS content. You can use a semitransparent surface to calibrate your position in world-scale AR.
- Scene view navigation constraint — By default, scene views constrain the camera to being above the ground. You should disable this feature to enable users to use world-scale AR underground (for example, while in a basement). The navigation constraint will interfere with tabletop AR if the user attempts to look at the scene from below.

To use `ARSceneView` , first add it to the view, then configure the lifecycle methods to start and stop tracking as needed.

```
using System;
using ARKit;
using Esri.ArcGISRuntime.ARToolkit;
using Foundation;
using UIKit;

namespace MyArApp
{
    public class BasicARExample : UIViewController
    {
        private ARSceneView _arSceneView;

        public override void LoadView()
        {
            // Create the views.
            View = new UIView();

            _arSceneView = new ARSceneView();
            _arSceneView.TranslatesAutoresizingMaskIntoConstraints = false;

            // Add the views.
            View.AddSubviews(_arSceneView);

            // Lay out the views.
            NSLayoutConstraint.ActivateConstraints(new[]{
                _arSceneView.TopAnchor.ConstraintEqualTo(View.TopAnchor),
                _arSceneView.BottomAnchor.ConstraintEqualTo(View.BottomAnchor),
```

Copyright © 1995-2020 Esri. All rights reserved.

```
                _arSceneView.LeadingAnchor.ConstraintEqualTo(View.LeadingAnchor),
                _arSceneView.TrailingAnchor.ConstraintEqualTo(View.TrailingAnchor)
            });
        }

        public override void ViewDidLoad()
        {
            base.ViewDidLoad();
            Initialize();
        }

        public override async void ViewDidAppear(bool animated)
        {
            base.ViewDidAppear(animated);
            await _arSceneView.StartTrackingAsync();
        }

        public override async void ViewDidDisappear(bool animated)
        {
            base.ViewDidDisappear(animated);
           if (_arSceneView != null)
           {
               await _arView.StopTrackingAsync()
           }
        }
    }
}
```

Configure privacy and permissions

Before you can use augmented reality, you'll need to request location and camera permissions.

On iOS, ensure the following properties are set in **info.plist** :

- `Privacy - Camera Usage Description`
- `Privacy - Location When In Use Usage Description`

The deployment target should be set to a supported version of iOS (see System requirements for details).

If you'd like to restrict your app to installing only on devices that support ARKit, add `arkit` to the required device capabilities section of **info.plist** :

```
<key>UIRequiredDeviceCapabilities</key>
<array>
    <string>arkit</string>
</array>
```

Once you have installed the toolkit, configured your app to meet privacy requirements, requested location permissions, and added an `ARSceneView` to your app, you can begin implementing your AR experience.

Understand Common AR Patterns

There are many AR scenarios you can achieve with Runtime. This SDK recognizes the following common patterns for AR:

- **Flyover** – Flyover AR is a kind of AR scenario that allows you to explore a scene using your device as a window into the virtual world. A typical flyover AR scenario will start with the scene's virtual camera positioned over an area of interest. You can walk around and reorient the device to focus on specific content in the scene.

Copyright © 1995-2020 Esri. All rights reserved.

- **Tabletop** – A kind of AR scenario where scene content is anchored to a physical surface, as if it were a 3D-printed model. You can walk around the tabletop and view the scene from different angles.
- **World-scale** – A kind of AR scenario where scene content is rendered exactly where it would be in the physical world. This is used in scenarios ranging from viewing hidden infrastructure to displaying waypoints for navigation. In AR, the real world, rather than a basemap, provides the context for your GIS data.

On screen, flyover is visually indistinguishable from normal scene rendering.

In tabletop, scene content is anchored to a real-world surface.

In world-scale AR, scene content is integrated with the real world.

Each experience is built using some combination of the features in Runtime and the toolkit and some basic behavioral assumptions.

AR pattern	Origin camera	Translation factor	Scene view	Base surface
Flyover AR	Above the tallest content in the scene	A large value to enable rapid traversal; 0 to restrict movement	Space effect: `Stars` Atmosphere: `Realistic`	Displayed
Tabletop AR	On the ground at the center or lowest point on the scene	Based on the size of the target content and the physical table	Space effect: None Atmosphere: None	Optional
World-scale AR	At the same location as the physical device camera	1, to keep virtual content in sync with real-world environment	Space effect: None Atmosphere: None	Optional for calibration

Note:
While the table describes the settings for the common AR scenarios in general, you have freedom to define your own configurations and build scenarios not explicitly mentioned in this guide.

Copyright © 1995-2020 Esri. All rights reserved.

Add tabletop AR to your app

Tabletop AR allows you to use your device to interact with scenes as if they were 3D-printed models sitting on your desk. You could, for example, use tabletop AR to virtually explore a proposed development without needing to create a physical model.

Note:
See Learn more below for links to additional resources.

Configure content for tabletop AR

The goal of a tabletop AR experience is to create the illusion of scene content sitting on your desk. Consider the following guidelines when creating content for your tabletop experience :

- Keep scene content focused; ideally a single geographic feature or a small area of a city.
- Ensure that scene content looks OK from below; your users should be able to look at the scene from an y angle.
- If you're not using a pre-defined scene (you allow users to load arbitrary scenes from ArcGIS Online, in other words), define a clipping distance to create a realistic tabletop experience.

Implement tabletop AR

Tabletop AR often allows users to place scene content on a physical surface (e.g. top of a desk) of their choice. Once the content is placed, it stays anchored to the surface as the user moves around it.

1. Create an `ARSceneView` and add it to the view
   See Enable your app for AR using AR Toolkit.
2. Listen for ARKit tracking tracking status changes on iOS and provide feedback to the user, for example when the user needs to move the phone more slowly or turn on more lights.

```
private void Initialize()
{
    // Rest of initialization ...

    // Listen for tracking status changes and provide feedback to the user.
    _arSceneView.ARSCNViewCameraDidChangeTrackingState += ARSceneView_TrackingStateChanged;
}

private void ARSceneView_TrackingStateChanged(object sender,
ARSCNViewCameraTrackingStateEventArgs e)
{
    // Provide clear feedback to the user in terms they will understand.
    switch (e.Camera.TrackingState)
    {
        case ARTrackingState.Normal:
            _arKitStatusLabel.Hidden = true;
            break;

        case ARTrackingState.NotAvailable:
            _arKitStatusLabel.Hidden = false;
            _arKitStatusLabel.Text = "ARKit location not available";
            break;

        case ARTrackingState.Limited:
            _arKitStatusLabel.Hidden = false;
            switch (e.Camera.TrackingStateReason)
            {
```

Copyright © 1995-2020 Esri. All rights reserved.

```
                case ARTrackingStateReason.ExcessiveMotion:
                    _arKitStatusLabel.Text = "Try moving your device more slowly.";
                    break;

                case ARTrackingStateReason.Initializing:
                    _arKitStatusLabel.Text = "Keep moving your device.";
                    break;

                case ARTrackingStateReason.InsufficientFeatures:
                    _arKitStatusLabel.Text = "Try turning on more lights and moving around.";
                    break;

                case ARTrackingStateReason.Relocalizing:
                    // This won't happen unless you enable relocalization.
                    break;
            }
            break;
    }
}
```

3. When tracking is ready and at least one plane has been found, wait for the user to tap. . You can use the `PlanesDetectedChanged` event to be notified when planes are detected.

```
public override void LoadView()
{
    // After creating and displaying views…
    // Get notification when planes are detected
    _arSceneView.PlanesDetectedChanged += ARSceneView_PlanesDetectedChanged;
}

private void ARSceneView_PlanesDetectedChanged(object sender, bool planeDetected)
{
    if (planeDetected)
    {
        BeginInvokeOnMainThread(EnableTapToPlace);
    }
}

private void EnableTapToPlace()
{
    // Show the help label.
    _helpLabel.Hidden = false;
    _helpLabel.Text = "Tap to place the scene.";

    // Wait for the user to tap.
    _arSceneView.GeoViewTapped += _arSceneView_GeoViewTapped;
}
```

4. Once the user has tapped a point, call `SetInitialTransformation`. The AR toolkit will use the native platform's plane detection to position the virtual camera relative to the plane. If the result is true, the transformation has been set successfully and you can place the scene.

```
private void _arSceneView_GeoViewTapped(object sender,
Esri.ArcGISRuntime.UI.Controls.GeoViewInputEventArgs e)
{
    if (_arSceneView.SetInitialTransformation(e.Position))
    {
        DisplayScene();
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    }
}
```

5. Create and display the scene. Set the navigation constraint on the scene's base surface to `None` . For demonstration purposes, this code uses the Philadelphia mobile scene package because it is particularly well-suited for tabletop display. You can download that .mspk and add it to your project to make the code below work. Alternatively, you can use any scene for tabletop mapping, but be sure to define a clipping distance for a proper tabletop experience.

```
private Scene _tabletopScene;

private async void DisplayScene()
{
    // Get the downloaded mobile scene package.
    MobileScenePackage package = await MobileScenePackage.OpenAsync("path_to_.mspk");
    await package.LoadAsync();
    _tabletopScene = package.Scenes.First();

    // Hide the base surface.
    _tabletopScene.BaseSurface.Opacity = 0;

    // Enable subsurface navigation. This allows you to look at the scene from below.
    _tabletopScene.BaseSurface.NavigationConstraint = NavigationConstraint.None;

    // Show the scene.
    _arSceneView.Scene = _tabletopScene;

    UpdateTranslationFactorAndOriginCamera(_tablestopScene);
}
```

6. Find an anchor point in the scene. You can use a known value, a user-selected value, or a computed value. For simplicity, this example uses a known value. Place the origin camera at that point.

```
private void UpdateTranslationFactorAndOriginCamera()
{
    // Create a camera at the bottom and center of the scene.
    //    This camera is the point at which the scene is pinned to the real-world surface.
    Camera originCamera = new Camera(39.95787000283599, -75.16996728256345, 8.813445091247559,
                                     0, 90, 0);

    // Set the origin camera.
    _arSceneView.OriginCamera = originCamera;
}
```

7. Set the translation factor on the ArcGIS AR view so that the whole scene can be viewed by moving around it. A useful formula for determining this value is **translation factor = virtual content width / desired physical content width**. The desired physical content width is the size of the physical table while virtual content width is the real-world size of the scene content; both measurements should be in meters. You can set the virtual content width by setting a clipping distance.

```
private void UpdateTranslationFactorAndOriginCamera()
{
    // Continued from above...

    // The width of the scene content is about 800 meters.
    double geographicContentWidth = 800;

    // The desired physical width of the scene is 1 meter.
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    double tableContainerWidth = 1;

    // Set the translation factor based on the scene content width and desired physical size.
    _arSceneView.TranslationFactor = geographicContentWidth / tableContainerWidth;
}
```

Guidelines for tabletop AR user experience

To create an optimal tabletop mapping experience:

- Provide user feedback for ARKit tracking issues. See Apple's Human Interface Guidelines for details.
- Set expectations about lighting before starting the experience. AR doesn't work well in low light.
- Don't allow users to view arbitrary content in AR. Use curated content that has been designed.
- Keep scene content focused; this should ideally be a single geographic feature or a small area of a city.
- Ensure that the scene looks good from below; your users may attempt to look at the scene from any angle.
- When using scenes that aren't confined to a small area, set a clipping distance to ensure that content is limited to a small area on the physical surface.

Learn more

- Apple: Human Interface Guidelines
- Sample: Display scenes in tabletop AR (iOS, Android)

Add flyover AR to your app

Flyover AR displays a scene while using the movement of the physical device to move the scene view camera. For example, you can walk around while holding up your device as a window into the scene. Unlike other AR experiences, the camera feed is not displayed to the user , making flyover more similar to a traditional virtual reality (VR) experience.

Flyover is the simplest AR scenario to implement, as there is only a loose relationship between the physical world and the rendered virtual world. With flyover, you can imagine your device as a window into the virtual scene.

Note:
See Learn more below for links to additional resources.

Implement flyover AR

1. Create the AR view and add it to the UI.
   See Add an AR view to the app.
2. Create the scene, add any content, then display it. This example uses an integrated mesh layer of an area on the US-Mexico border.

```
private void DisplayScene()
{
    // Create the scene with a basemap.
    Scene flyoverScene = new Scene(Basemap.CreateImagery());

    // Create the integrated mesh layer and add it to the scene.
    IntegratedMeshLayer meshLayer = new IntegratedMeshLayer(new
```

Copyright © 1995-2020 Esri. All rights reserved.

```
System.Uri("https://www.arcgis.com/home/item.html?id=dbc72b3ebb024c848d89a42fe6387a1b"));
    flyoverScene.OperationalLayers.Add(meshLayer);

    // Show the scene.
    _arSceneView.Scene = flyoverScene;

    // TODO – configure origin camera for AR
}
```

3. Place the origin camera above the content you want the user to explore, ideally in the center. Typically, you'll want to place the origin camera above the highest point in your scene.

```
private void DisplayScene()
{
    // Continued from above
    // Wait for the layer to load so that extent is available.
    await meshLayer.LoadAsync();

    // Start with the camera at the center of the mesh layer.
    Envelope layerExtent = meshLayer.FullExtent;
    Camera originCamera = new Camera(layerExtent.GetCenter().Y, layerExtent.GetCenter().X,
600, 0, 90, 0);
    _arSceneView.OriginCamera = originCamera;
}
```

4. Set the translation factor to allow rapid traversal of the scene. The translation factor defines the relationship between physical device movement and virtual camera movement. To create a more immersive experience, set the space effect on the scene view to `Stars` and the atmosphere effect to `Realistic` . Disable the navigation constraint on the scene's base surface to prevent camera position problems near the ground.

```
private void DisplayScene()
{
    // Continued from above

    // Disable navigation constraint
    _scene.BaseSurface.NavigationConstraint = NavigationConstraint.None;

    // Set the translation factor to enable rapid movement through the scene.
    _arSceneView.TranslationFactor = 1000;

    // Enable atmosphere and space effects for a more immersive experience.
    _arSceneView.SpaceEffect = SpaceEffect.Stars;
    _arSceneView.AtmosphereEffect = AtmosphereEffect.Realistic;
}
```

Guidelines for flyover AR

Consider the following guidelines for creating high-quality flyover AR experiences:

- Provide actionable feedback to the user when there are ARKit tracking issues. Tracking problems caused by environmental factors like low light can break the AR experience.
- Set expectations about the environment before starting the AR experience. Flyover AR doesn't work well in low light or in tight, constrained spaces.

Copyright © 1995-2020 Esri. All rights reserved.

- Start the AR experience in the center of the area of interest for your scene. Users are likely to move around freely, including rotating and looking behind where they originally start.

Learn more

- Apple: Human Interface Guidelines
- Apple: ARSCNViewDelegate
- Sample: Explore scenes in flyover AR (iOS, Android)

Add world-scale AR to your app

A world-scale AR experience is defined by the following characteristics:

- The scene camera is positioned to precisely match the position and orientation of the device's physical camera
- Scene content is placed in the context of the real world by matching the scene view's virtual camera position and orientation to that of the physical device camera.
- Context aids, like the basemap, are hidden; the camera feed provides real-world context.

Some example use cases of world-scale AR include:

- Visualizing hidden infrastructure, like sewers, water mains, and telecom conduits.
- Maintaining context while performing rapid data collection for a survey.
- Visualizing a route line while navigating.

Note:
See Learn more below for links to additional resources.

Configure content for world-scale AR

The goal of a world-scale AR experience is to create the illusion that your GIS content is physically present in the world around you. There are several requirements for content that will be used for world-scale AR that go beyond what is typically required for 2D mapping.

- Ensure that all data has an accurate elevation (or Z) value. For dynamically generated graphics (for example, route results) use an elevation surface to add elevation.
- Use an elevation source in your scene to ensure all content is placed accurately relative to the user.
- Don't use 3D symbology that closely matches the exact shape of the feature it represents. For example, do not use a generic tree model to represent tree features or a fire hydrant to represent fire hydrant features. Generic symbology won't capture the unique geometry of actual real-world objects and will highlight minor inaccuracies in position.
- Consider how you present content that would otherwise be obscured in the real world, as the parallax effect can make that content appear to move unnaturally. For example, underground pipes will 'float' relative to the surface, even though they are at a fixed point underground. Have a plan to educate users, or consider adding visual guides, like lines drawn to link the hidden feature to the obscuring surface (for example, the ground).
- By default, ArcGIS Runtime renders content over a large distance, which can be problematic when you are trying to view a limited subset of nearby features (just the pipes in your building, not for the entire campus, for example). You can use the clipping distance to limit the area over which scene content renders.

Copyright © 1995-2020 Esri. All rights reserved.

Understand location tracking options for world-scale AR

There are a few strategies for determining the device's position in the world and maintaining that position over time:

- Use the device's location data source (for example, GPS) to acquire an initial position and make further position updates using ARKit and ARCore only.
- Use the location data source continuously.

With continuous updates, the origina camera is set every time the location data source provides a new update. With a one-time update, the origin camera is set only once.

There are benefits and drawbacks to each approach that you should consider when designing your AR experience:

- One-time update
  - Advantage: ARKit/ARCore tracking is more precise than most location data sources.
  - Advantage: Content stays convincingly pinned to its real-world position, with minimal drifting or jumping.
  - Disadvantage: Error accumulates the further you venture from where you start the experience.
- Continuous update
  - Advantage: Works over a larger area than ARKit or ARCore.
  - Disadvantage: Visualized content will jump as you move through the world and the device's location is updated (as infrequently as once per second rather than ARKit's 60 times per second).
  - Disadvantage: Because the origin camera is constantly being reset, you can't use panning to manually correct position errors.

You don't need to make a binary choice between approaches for your app. Your app can use continuous updates while the user moves through larger areas, then switch to a primarily ARKit or ARCore-driven experience when you need greater precision.

With `ARSceneView`, the choice of location strategy is specified in the call to `StartTrackingAsync`. To change the location update mode, stop tracking and then resume tracking with the desired mode.

Note:
In the samples, continuous update mode is referred to as **Roaming** and one-time update mode is referred to as **Local**.

Implement world-scale AR

1. Create an `ARSceneView` and add it to the view.
   See Add an AR view to your app.
2. Configure the `ARSceneView` with a location data source. The location data source provides location information for the device. The AR scene view uses the location data source to place the virtual scene camera close to the location of the physical device's camera.
   ```
   _arView.LocationDataSource = new SystemLocationDataSource();
   ```
3. Configure the scene for AR by setting the space and atmosphere effects and adding an elevation source, then display it.
   ```
   private void ConfigureSceneForAR()
   {
       // Create the scene with imagery basemap
   ```

Copyright © 1995-2020 Esri. All rights reserved.

```
    _arView.Scene = new Scene(Basemap.CreateImagery ());

    // Create and add the elevation surface
    _elevationSource = new ArcGISTiledElevationSource(new Uri("https://elevation3d.arcgis.com/
arcgis/rest/services/WorldElevation3D/Terrain3D/ImageServer"));
    _elevationSurface = new Surface();
    _elevationSurface.ElevationSources.Add(_elevationSource);
    _arView.Scene.BaseSurface = _elevationSurface;

    // Remove the navigation constraint.
    _elevationSurface.NavigationConstraint = NavigationConstraint.None;

    // Disable space and atmosphere effects
    _arView.SpaceEffect = SpaceEffect.None;
    _arView.AtmosphereEffect = AtmosphereEffect.None;
}
```

4. Start tracking using one of two location update modes, continuous or once only.

```
public override void ViewDidAppear(bool animated)
{
    base.ViewDidAppear(animated);

    // Continuous update mode
    _arView.StartTrackingAsync(ARLocationTrackingMode.Continuous);

    // One-time mode
    //_arView.StartTrackingAsync(ARLocationTrackingMode.Initial);
}
```

5. Provide a calibration UI to allow your users to correct heading, elevation, and location errors. See Enable calibration for world-scale AR for more details.

Learn more

- Apple: Human Interface Guidelines
- Apple: ARSCNViewDelegate
- Sample: Navigate in AR (iOS, Android)

Guidelines for world-scale AR

Consider the following guidelines to help you overcome common obstacles to creating a useful world-scale AR experience:

- Create a calibration experience that works for your users' needs in the environment they will be working in. See Enable calibration for world-scale AR for details.
- Educate your users on what to expect from world-scale experiences, including calibration and parallax issues.
- Consider how and where your users will use your AR experience. Due to limitations in GNSS and compass accuracy, your users will need to calibrate their initial starting position. Ensure that your users have the tools they need to calibrate their position, whether that is a known landmark, a fixed calibration point, or comparison to known reference features. See Enable calibration for world-scale AR for details.
- Consider encouraging users to use high-accuracy positioning devices if practical.

Copyright © 1995-2020 Esri. All rights reserved.

Enable calibration for world-scale AR

World-scale AR depends on a close match between the positions and orientations of the device's physical camera and the scene view's virtual camera. Any error in the device's position or orientation will degrade the experience. Consider each of the following key properties as common sources of error:

- **Heading** – Usually determined using a magnetometer (compass) on the device
- **Elevation/Altitude (Z)** – Usually determined using GPS/GNSS or a barometer
- **Position (X,Y)** – usually determined using GPS/GNSS, cell triangulation, or beacons

The following examples illustrate these errors by showing a semitransparent basemap for comparison with the ground truth provided by the camera:

Orientation error

Elevation error

Position error

Design a calibration workflow

There are many ways to calibrate position and heading. In most scenarios, you'll need to provide one or more points of comparison between scene content and the real-world ground truth. Consider the following options for allowing the user to visually calibrate the position:

- Align the imagery on the basemap with the camera feed.
- Align a known calibration feature with its real-world equivalent (for example, a previously recorded tree feature).
- Define a start point and heading and direct the user.

Consider the following UI options for allowing the user to adjust the calibration:

- Display sliders for orientation and elevation adjustment.
- Use 'joystick' sliders, where the further from the center the slider moves, the faster the adjustment goes.
- Use an image placed in a known position in conjunction with ARKit / ARCore image detection to automatically determine the device's position.

Explicitly plan for calibration when designing your AR experiences. Consider how and where your users will use your app. Not all calibration workflows are appropriate for all locations or use cases.

Note:

As described in Understand two location tracking options for world-scale AR , you can configure the AR view to reset the device's position using the location data source continuously or just once. If you would like to manually adjust position or altitude while using continuous location updates, you will need to implement a custom location data source.

Copyright © 1995-2020 Esri. All rights reserved.

Identify real-world and in-scene objects

ArcGIS Runtime scene views have two methods for determining the location in a scene that corresponds to a point on the device's screen:

- `ScreenToBaseSurface` – ignores non-surface content, like 3D buildings
- `ScreenToLocationAsync` – includes non-surface content

`ARSceneView` has `ARScreenToLocation` , which:

1. Performs a hit test using ARKit / ARCore to find a real- world plane.
2. Applies a transformation to determine the physical position of that plane relative to the known position of the device's camera.
3. Returns the real- world position of the tapped plane.

You can use `ARScreenToLocation` to enable field data collection workflows where users tap to identify real-world objects in the camera feed as detected by ARKit / ARCore . The position of the tapped object will be more accurate than using the device's location, as you might with a typical field data collection process.

Manage vertical space in world-scale AR

Accurate positioning is particularly important to world-scale AR; even small errors can break the perception that the virtual content is anchored in the real world. Unlike 2D mapping, Z values are important. And unlike traditional 3D experiences, you need to know the position of the user’s device.

Be aware of the following common Z-value challenges that you’re likely to encounter while building AR experiences:

- **Many kinds of Z values** – Android and iOS devices differ in how they represent altitude/elevation/Z values.
- **Imprecise altitude** – Altitude/Elevation is the least precise measurement offered by GPS/GNSS. In testing, we found devices reported elevations that were anywhere between 10 and 100 above or below the true value, even under ideal conditions.

Many kinds of Z values

Just as there are many ways to represent position using X and Y values, there are many ways to represent Z values. GPS devices tend to use two primary reference systems for altitude/elevation:

- WGS84 – Height Above Ellipsoid (HAE)
- Orthometric – Height Above Mean Sea Level (MSL)

The full depth of the differences between these two references is beyond the scope of this topic, but do keep in mind the following facts:

- Android devices return elevations in HAE, while iOS devices return altitude in MSL.
- It is not trivial to convert between HAE and MSL; MSL is based on a measurement of the Earth’s gravitational field. There are many models, and you may not know which model was used to when generating data.
- Esri’s world elevation service uses orthometric altitudes.
- The difference between MSL and HAE varies by location and can be on the order of tens of meters. For example, at Esri’s HQ in Redlands, California, the MSL altitude is about 30 meters higher than the HAE elevation.

Copyright © 1995-2020 Esri. All rights reserved.

It is important that you understand how your Z values are defined to ensure that data is placed correctly in the scene. For example, the Esri world elevation service uses MSL for its Z values. If you set the origin camera using an HAE Z value, you could be tens of meters off from the desired location.

To gain a deeper understanding of these issues, see ArcUser: Mean Sea Level, GPS, and the Geoid.

Learn more

- ArcUser: Mean Sea Level, GPS, and the Geoid
- GPS.gov: GPS Accuracy - learn about GPS accuracy and limitations to vertical accuracy

Visualize planes and features detected by ARKit and ARCore

Some workflows, like tapping to place a tabletop scene or collecting a feature, rely on ARKit / ARCore features that detect planes. Plane visualization is particularly useful for two common scenarios:

- Visualization provides feedback to users, so they know which surfaces the app has detected and can interact with
- Visualization is useful while developing and testing your app

ARKit / ARCore can be configured to render planes.

```
_arSceneView.RenderPlanes = true;
```

Copyright © 1995-2020 Esri. All rights reserved.

Offline with maps and scenes

Copyright © 1995-2020 Esri. All rights reserved.

Work offline

Offline maps and scenes allow users to continue being productive when network connectivity is poor or nonexistent. Users can explore maps, collect information, edit their asset data, find places, and routes to new locations, all while disconnected. Once a connection is reestablished, users can update their offline map by downloading changes relevant to their offline map or by synchronizing their edits with the online services. Users can also explore scenes, manage camera controllers, and find locations, all while disconnected.

ArcGIS Runtime supports maps that are always connected, occasionally connected, and fully disconnected.

- Always connected — Users expect their apps to have constant access to online map and layer services. If the device loses network connectivity, the application will lose access to these services, resulting in loss of data and, perhaps, app failure.
- Occasionally connected—Apps that are occasionally connected can take maps offline when a connection is available and continue to work offline when the device is disconnected. When connectivity is restored, any changes to operational data can be synchronized with the online services. There following options support occasionally connected workflows:
  - Take a map offline - preplanned—The map author defines a map area and generates the offline map ahead of time so that any number of field workers can download the same map and take it into the field.
  - Take a map offline - on-demand—The field worker defines the map area, generates the offline map content, and downloads the map to their device.

  You must enable your online services for offline use to support both of these workflows.
- Fully disconnected: Apps can operate in a fully disconnected environment by opening mobile map packages or mobile scene packages created with ArcGIS Pro. These read-only packages can be shared within an organization or distributed by traditional means and copied onto any number of devices. This fully disconnected workflow is suitable for read-only apps that do not require regular data updates. See Take a map offline - ArcGIS Pro or Take a scene offline - ArcGIS Pro for more details.

Workflow capabilities summary

Capability	Preplanned (map)	On-demand (map)	ArcGIS Pro (map)	ArcGIS Pro (scene)
Display, interact, identify, and analyze layers.	Yes	Yes	Yes	Yes
Edit feature data in the mobile geodatabase.	Yes	Yes	No	No
Synchronize edits with the online feature service.	Yes	Yes	No	No
Share offline maps or scenes with other users.	Yes	No	No	No
Receive scheduled updates from feature services.	Yes	No	No	No
Geocoding.	No	No	Yes	Yes
Routing.	No	No	Yes	No
Support raster layers.	No	No	Yes	Yes
Supports online layers and tables	No	No	Yes	Yes

An alternative to taking a map offline is to construct the map using individual offline layers. Layers can either reference offline data sources on the device or be exported from suitable online services. Working directly with layers gives you full flexibility to compose your own offline map. For more information, see Work with offline layers and Take a layer offline.

Copyright © 1995-2020 Esri. All rights reserved.

Enable services for offline use

If you adopt the preplanned or on-demand workflows, you must ensure that all ArcGIS Online or Enterprise services contained in the web map can be taken offline. The author of the web map can set these options when publishing, or later using any of the service management tools.

Feature services

You must sync-enable the feature service if you want the feature data to be downloaded with the offline map, scheduled updates to be received, and any edits synchronized. Follow the instructions here:

- In the ArcGIS Enterprise help, see Prepare data for offline use in the ArcGIS Server documentation.
- In the ArcGIS Online help, see Manage editor settings.

Raster and vector tiled services

You must ensure that tiled services have the `exportTiles` capability enabled to allow tiled services data to be exported and downloaded with the offline map. Follow the instructions here:

- In the ArcGIS Enterprise help, see Allow clients to export cache tiles.
- In the ArcGIS Online help, see Manage hosted vector tile layer settings.

Copyright © 1995-2020 Esri. All rights reserved.

Take a map offline - on-demand

The on-demand workflow allows your field workers to define an area of a map and control the map content to be taken offline. Your app can specify parameters and overrides to control the map's area of interest, max and min scale, filter features, include attachments, determine editing characteristics, and many other aspects of the map content. The aim of these parameters is to ensure that only essential map content is taken offline. Here are the steps for this on-demand workflow:

1. Create an offline map task
2. Examine the map's offline capabilities
3. Create parameters to specify the map content
4. Create a job to generate an offline map
5. Run the job

Note:
To take a map offline on-demand, use ArcGIS Online or ArcGIS Enterprise 10.3 or later. You must ensure that all services used in the web map are enabled for offline use.

Create an offline map task

Create an `OfflineMapTask` from either an online map or its portal item.

```
// Get a web map item using its ID.
PortalItem webmapItem = await PortalItem.CreateAsync(portal, "acc027394bc84c2fb04d1ed317aac674");

// Create a map from the web map item.
Map onlineMap = new Map(webmapItem);

// Create an OfflineMapTask from the map ...
takeMapOfflineTask = await OfflineMapTask.CreateAsync(onlineMap);
// ... or a web map portal item.
takeMapOfflineTask = await OfflineMapTask.CreateAsync(webmapItem);
```

Examine the map's offline capabilities

We recommend that you check that all the layers and tables in the map can be taken offline by examining the map's offline capabilities. You should be aware of which layers or tables are missing since the map may not be as complete as originally designed.

Obtain the `OfflineMapCapabilites` object by calling the `GetOfflineMapCapabilitiesAsync` method on the offline map task. The `Error` boolean value will indicate whether at least one layer or table cannot be taken offline. Examine the `LayerCapabilities` and `TableCapabilities` to determine which layer or table cannot be taken offline along with the reason why.

```
OfflineMapCapabilities results = await task.GetOfflineMapCapabilitiesAsync(parameters);
if (results.HasErrors)
{
    // Handle possible errors with layers
    foreach (var layerCapability in results.LayerCapabilities)
    {
        if (!layerCapability.Value.SupportsOffline)
```

Copyright © 1995-2020 Esri. All rights reserved.

```
        {
            Debug.WriteLine(layerCapability.Key.Name + " cannot be taken offline. Error : " +
layerCapability.Value.Error.Message);
        }
    }

    // Handle possible errors with tables
    foreach (var tableCapability in results.TableCapabilities)
    {
        if (!tableCapability.Value.SupportsOffline)
        {
            Debug.WriteLine(tableCapability.Key.TableName + " cannot be taken offline. Error : "
+ tableCapability.Value.Error.Message);
        }
    }
}
else
{
    // All layers and tables can be taken offline!
    ShowMessage("All layers are good to go!");
}
```

Note:
Although this step is optional, we recommend that you examine these capabilities to ensure that the map contains the appropriate layers and tables.

Create parameters to specify map content

When you take a map offline you want the map to contain all the content that is relevant to the field worker. Be aware, however, that the size of the map will directly impact the time it takes to generate the offline map and download it to your device. You can control this size by specifying the map's area of interest to cover just the geographical area that the field worker needs.

1. Create the `GenerateOfflineMapParameters` by passing an area of interest to the `CreateDefaultGenerateOfflineMapParametersAsync` method on the `OfflineMapTask`.
2. Obtain and examine the returned parameters. These default parameters represent the advanced offline settings configured by the web map author.
3. To override any of these default parameters see Advanced parameters. For example you can, set the max and min scale, use a local basemap, or set a definition expression on features.

```
// Create default parameters for the task.
Envelope areaOfInterest = GetAreaOfInterest();
GenerateOfflineMapParameters parameters = await
takeMapOfflineTask.CreateDefaultGenerateOfflineMapParametersAsync(areaOfInterest);

// Limit the maximum scale to 5000 but take all the scales above (use default of 0 as the
MinScale).
parameters.MaxScale = 5000;

// Set attachment options.
parameters.AttachmentSyncDirection = AttachmentSyncDirection.Upload;
parameters.ReturnLayerAttachmentOption = ReturnLayerAttachmentOption.EditableLayers;

// Request the table schema only (existing features won’t be included).
```

Copyright © 1995-2020 Esri. All rights reserved.

```
parameters.ReturnSchemaOnlyForEditableLayers = true;

// Update the map title to contain the region.
parameters.ItemInfo.Title = parameters.ItemInfo.Title + " (Central)";

// Override the thumbnail with a new image based on the extent.
RuntimeImage thumbnail = await MyMapView.ExportImageAsync();
parameters.ItemInfo.Thumbnail = thumbnail;
```

Note:

Instead of obtaining the default set of `GenerateOfflineMapParameters` you could construct them manually. The author of the web map can recommend how data is downloaded to offline devices, and subsequently updated, by adjusting the map's advanced offline settings. See Take web maps offline. You can access these advanced settings from the map's `OfflineSettings`. They provide options for:

- working with feature attachments
- synchronizing edits
- downloading a basemap or using one which is already on the device

Create a job to generate an offline map

To generate and download the offline map you must create a `GenerateOfflineMapJob` by providing the `GenerateOfflineMapParameters` to the `GenerateOfflineMap` method on `OfflineMapTask`. You must provide a directory on the device to store the offline map. If this download directory already exists it must be empty. If the directory doesn't exist, it will be created by the job.

```
// Create the job to generate an offline map, pass in the parameters and a path to store the map
package.
GenerateOfflineMapJob generateMapJob = takeMapOfflineTask.GenerateOfflineMap(parameters,
pathToOutputPackage);
```

If you want to control the individual layer and table content you also need to provide the `GenerateOfflineMapParameterOverrides` as well as the `GenerateOfflineMapParameters` to the `GenerateOfflineMap` method on `OfflineMapTask`. For more details see Create offline map parameter overrides.

```
// Create the job to generate an offline map, pass in the parameters, a path to store the map
package, and overrides.
GenerateOfflineMapJob offlineMapJob = takeMapOfflineTask.GenerateOfflineMap
    (parameters, pathToOutputPackage, offlineMapParamOverrides);
```

Note:

Note that the job will behave the same whether it was created using just the `GenerateOfflineMapParameters` or using both the `GenerateOfflineMapParameterOverrides` along with the `GenerateOfflineMapParameters`.

See the Tasks and jobs topic for more details on how to work with jobs in general.

Copyright © 1995-2020 Esri. All rights reserved.

Run the job

To generate the offline map and download it to your device, start the `GenerateOfflineMapJob` . Upon completion the job will return an instance of the `GenerateOfflineMapResult`. If at least one table or layer failed to be taken offline the `HasErrors` property will be `true`. In this case you should examine the `layerErrors` and `tableErrors` dictionaries to identify the problem.

Remember, that if the job's `ContinueOnErrors` parameter has been set to false then the job will be terminated immediately if a single layer or table fails to be taken offline.

If you want to display the map immediately, pass the `GenerateOfflineMapResult.OfflineMap` to the map view.

```
// Generate the offline map and download it.
GenerateOfflineMapResult offlineMapResult = await offlineMapJob.GetResultAsync();

if (!offlineMapResult.HasErrors)
{
    // Job completed successfully and all content was generated.
    Debug.WriteLine("Map " +
        offlineMapResult.MobileMapPackage.Item.Title +
        " was saved to " +
        offlineMapResult.MobileMapPackage.Path);

    // Show the offline map in a MapView.
    MyMapView.Map = offlineMapResult.OfflineMap;
}
else
{
    // Job is finished but one or more layers or tables had errors.
    if (offlineMapResult.LayerErrors.Count > 0)
    {
        // Show layer errors.
        foreach (var layerError in offlineMapResult.LayerErrors)
        {
            Debug.WriteLine("Error occurred when taking " +
                layerError.Key.Name +
                " offline. Error : " +
                layerError.Value.Message);
        }
    }
    if (offlineMapResult.TableErrors.Count > 0)
    {
        // Show table errors.
        foreach (var tableError in offlineMapResult.TableErrors)
        {
            Debug.WriteLine("Error occurred when taking " +
                tableError.Key.TableName +
                " offline. Error : " +
                tableError.Value.Message);
        }
    }
}
```

Offline maps created by the on-demand workflow are stored in an unpacked mobile map package. When your app goes into the field, you will need to open the map directly from this mobile map package `downloadDirectory` stored on your device. See Open an offline map for more information.

Copyright © 1995-2020 Esri. All rights reserved.

Advanced parameters

Once you have obtained the `GenerateOfflineMapParameters`, you can set any of their properties to control the offline map content. For example:

- The map's max and min scale
- Whether basemaps are included in the map
- Whether to apply definition expression filters
- Whether to return all rows from a related table
- Whether to continue downloading the map if a single layer fails to be taken offline
- Including feature attachments
- Whether only the schema is provided for editable layers
- Update or replace the map's metadata

Note:
If your use-case requires you to have more control over how individual layers and tables are downloaded, you can use the `GenerateOfflineMapParameterOverrides` to make those specific adjustments. Follow the steps outlined in Create offline map parameter overrides.

Scale range

Tiled layers are comprised of many tiles at different levels of detail. For a particular area of interest, the amount of space, generation time, and download time will increase significantly with every increased level of detail. It's strongly recommended that you only download the levels of detail that are relevant for your app. You can control this by setting the minimum and maximum scale parameters (`MinScale` and `MaxScale`). If possible, choose a maximum scale that is not too “zoomed in”, as this will generate a large number of unnecessary tiles. Each service has a limit to the number of tiles which can be taken offline. It's recommended that you always set a scale range to avoid hitting this limit.

Note:
On-demand workflow will honor the scale range of a tiled layer as specified by the web map. This behavior helps to ensure that only tile data required by the offline map will be downloaded.

For example, if the web map contains a tiled layer which is set to a min scale of 1:1,000,000 and a max scale of 1:10,000, these values will be applied to the scale range requested by the user. If a user requests the data with a scale range of 1:900,000 to 1:5,000, the minimum range of 1:900,000 (min scale defined in the user's parameters) to 1:10,000 (max scale defined on the web map) will be downloaded. If the user's parameters request the default scale values of 1:0, then the range from the web map will be applied.

Include a map's basemap

The author of the web map can define whether the map:

- The basemap defined by the web map
  This is the default situation and ensures that a tile package is downloaded as part of the mobile map package.
- A tile package that is already on the device

Copyright © 1995-2020 Esri. All rights reserved.

The tile package must be downloaded or copied onto the device and can be located using an actual file path on the device or a path that is relative to the mobile map package. You must ensure that the tile package covers the areas required by your map area. The benefits of this option are that the mobile map package file will be smaller, the download time may be faster, and you can use the tile package in many different maps and apps.

To use the tile package on your device you must set the `referenceBasemapDirectory` to the directory which contains the tile package. We recommend that you confirm that the tile package file, `referenceBasemapFilename`, exists on the device before running the `AGSGenerateOfflineMapJob`. This job will add the tile package, as a basemap, to the offline map.

Note:
If this tile package has a different spatial reference to your online map then the offline map will use the spatial of the tile package.

- No basemap
  If only want to take the operational layers offline you can programmatically exclude the basemap. To do this set the `includeBasemap` property to false. In this case the `GenerateOfflineMapJob` will not download any layers included as part of the map's `basemap`. This task will not use the local tile package, even if you have specified one.

Apply feature layer definition expression filters

Whilst taking a map offline the `GenerateOfflineMapJob` will apply the feature layer's definition expression by default. Applying the definition expression may reduce the number of features taken offline for display and sync. If you do not want to apply the definition expression then set the boolean value of `IsDefinitionExpressionFilterEnabled` to be `false`.

Return all rows from a related table

If a map contains a layer or table that has a relationship with another table then you can choose to download all, or only the related rows from the destination table. The default is to download only the related rows. If you want to return all rows then you must set the value of the `destinationTableRowFilter` to be .

If the table is not the direct destination of a feature table, or it is a standalone table, or it is the source of a relationship then all rows are returned.

For more information, see Essentials of relating tables.

Continue downloading the map if a single layer or table fails

By default the `GenerateOfflineMapJob` will continue to take layers and tables offline even if a layer or table has failed. Whilst this ensures that the map is taken offline there can be missing data. When this job completes you should examine the job's result (discussed under Run the job) to identify if and why any layers have failed. At this point you can decide whether to continue working with the map.

If you want the job to stop immediately, if a layer or table fails to download, then set the `ContinueOnErrors` property to be `false`. In this case if a map is successfully taken offline it will contain all of its layers and tables.

The failure to take a layer or table offline may be due to an intermittent network connection, loss of the service or an unsupported layer type.

Copyright © 1995-2020 Esri. All rights reserved.

Inclusion of feature attachments

Some feature services can add attachments (pictures, videos, and other documents) to individual features. Since these files can be large, you should consider your app's offline workflow to determine whether the attachments are needed by the app, and whether attachments need to be synchronized with the service when the app is next online. These two behaviors work in conjunction with each other, and are defined using the `ReturnLayerAttachmentOption` and `AttachmentSyncDirection` properties on the `GenerateOfflineMapParameters` class.

- The return layer attachment property defines which layers should contain attachments in the offline map. The options are:
  - `None` - None of the layers will contain any attachments.
  - `AllLayers` - All layers will have their attachments included.
  - `ReadOnlyLayers` - Layers without editing enabled will have attachments included.
  - `EditableLayers` - Layers with editing enabled will have attachments included.
- The attachment sync direction defines how the attachments are synchronized with the service. The options are:
  - `None` - Attachments are not synchronized as part of the synchronization operation.
  - `Upload` - Attachments are uploaded from the client to the service, but any changes on the service are not downloaded to the client.
  - `Bidirectional` - Attachments are uploaded from client to the service, and changes on the service are pulled down to the client.

Note:
Attachment sync direction was introduced with ArcGIS Server 10.4.

Inclusion of features from editable feature layers

Here are some workflows that describe how these two parameters affect each other.

- Workflow 1—Download attachments for all layers in the map, allow the user to add or remove attachments from the layers, and then synchronize these changes between the service and the client when online. For example: multiple users collect data on the same area and they want to synchronize all the changes with the centralized services as well as sharing changes with other people in the field.
  - `ReturnLayerAttachmentOption.AllLayers`
  - `AttachmentSyncDirection.Bidirectional`
- Workflow 2—Download attachments for all read-only layers and update these layers when online. For example: users are offline and viewing a layer of buildings with photos that show how the buildings look. If there are any new photos added to the service, these will be downloaded to the client during synchronization when online.
  - `ReturnLayerAttachmentOption.ReadOnlyLayers`
  - `AttachmentSyncDirection.Bidirectional`
- Workflow 3—Download attachments for editable layers only and upload them to the service when online. For example: users are offline and only need to view attachments for editable layers. If there are any read-only layers that provide context for the map, their attachments aren't included to the local map. If users remove or add any new attachments, these changes can be synchronized to the service when online.
  - `ReturnLayerAttachmentOption.EditableLayers`

Copyright © 1995-2020 Esri. All rights reserved.

  - `AttachmentSyncDirection.Bidirectional`
- Workflow 4—Do not download any attachments but allow any new attachments to be uploaded to the service when online. For example: users are offline and collecting new attachments in the field but do not need to view existing attachments.
  - `ReturnLayerAttachmentOption.None`
  - `AttachmentSyncDirection.Upload`

If users are collecting new information in the field where they do not need to access previously created features, you can create an offline map with empty editable feature layers. Do this by setting the `GenerateOfflineMapParameters` property `ReturnSchemaOnlyForEditableLayers` to `true`.

Update or replace map's metadata

You can access the online map's metadata from the `ItemInfo` property. It includes portal item properties such as the title, description, short description, and thumbnail. This information is populated from the portal item that contains the map. You can override any of these metadata properties before you take the map offline. For example, if you are creating offline maps of different areas of interest on the same map, you may want to change the map's title to indicate which area it contains.

You can also create a new `OfflineMapItemInfo` object and manually set all the details.

```
// Create a new OfflineMapItemInfo to store metadata for the map.
OfflineMapItemInfo itemInfo = new OfflineMapItemInfo();

// Create new thumbnail from the map
RuntimeImage thumbnailImage = await MyMapView.ExportImageAsync();

// Provide a better title and thumbnail to describe the map extent.
itemInfo.Thumbnail = thumbnailImage;
itemInfo.Title = "Water network (Central)";

// Copy basic info from the source item.
itemInfo.Snippet = webmapItem.Snippet;
itemInfo.Description = webmapItem.Description;
itemInfo.AccessInformation = webmapItem.AccessInformation;

// Add some tags.
itemInfo.Tags.Add("Water network");
itemInfo.Tags.Add("Data validation");

// Apply the metadata to the offline map parameters.
parameters.ItemInfo = itemInfo;
```

Create offline map parameter overrides

Managing the map content may be sufficient for your workflow but in some cases you may also want to control how individual layers or tables are taken offline, such as the following:

- Reducing the amount of data (for example, tile data) for a given layer
- Altering the spatial extent of a given layer (for example, to give coverage beyond the study area)
- Filtering features (for example, with a where clause) to only take those that are relevant to your fieldwork
- Taking features with null geometry (for example, where the attributes are populated in the office but the geometry needs to be captured on site)
- Omitting individual layers

Copyright © 1995-2020 Esri. All rights reserved.

If you need this fine-grained control then you must generate the `GenerateOfflineMapParameterOverrides` object. This object gives you access to three dictionaries containing the generate geodatabase parameters, export tile cache parameters and export vector tile parameters. Adjust any of these parameters and create the `GenerateOfflineMapJob` using the overrides object. To control the individual layers and tables that you take offline follow these steps:

1. Generate and modify the `GenerateOfflineMapParameters` as described in Create offline map parameter overrides above.
2. Generate the parameter overrides object (`GenerateOfflineMapParameterOverrides`) using the `generateOfflineMapParameterOverrides` method on the `OfflineMapTask`. Provide the `GenerateOfflineMapParameters` generated from the previous step.
3. When the task completes, the `GenerateOfflineMapParameterOverrides` will be initialized, honoring the settings in the supplied `GenerateOfflineMapParameters`. `GenerateOfflineMapParameterOverrides` will present three sets of data-source parameters as dictionary structures:
   - GenerateGeodatabaseParameters
   - ExportTileCacheParameters
   - ExportVectorTileCacheParameters

   You may inspect the dictionaries and modify them before they are passed on to the job to download the map. Each dictionary contains key, value pairs where the key is an `OfflineMapParametersKey` and the value is a data-source specific parameter (for example, a `GenerateGeodatabaseParameters`).

```
// Get the offline map parameter overrides from the map task (pass in the task parameters).
GenerateOfflineMapParameterOverrides offlineMapParamOverrides =
    await takeMapOfflineTask.CreateGenerateOfflineMapParameterOverridesAsync(parameters);

// Get the dictionary of parameter overrides for exporting a tile cache.
IDictionary<OfflineMapParametersKey, ExportTileCacheParameters> tileCacheParamOverrides =
    offlineMapParamOverrides.ExportTileCacheParameters;
// Get the dictionary of parameter overrides for exporting vector tiles.
IDictionary<OfflineMapParametersKey, ExportVectorTilesParameters> vectorTileParamOverrides =
    offlineMapParamOverrides.ExportVectorTilesParameters;
// Get the dictionary of parameter overrides for generating a geodatabase.
IDictionary<OfflineMapParametersKey, GenerateGeodatabaseParameters> geodatabaseParamOverrides
=
    offlineMapParamOverrides.GenerateGeodatabaseParameters;
// TODO: Override parameter values as needed.
```

4. If, however, you just want to look up a specific set of parameters for a specific layer or table, you can construct the key from the layer or the table and obtain the parameters from the dictionary.

```
// Create a dictionary key for the desired parameter override.
OfflineMapParametersKey streetsParameterKey = new OfflineMapParametersKey(streetsLayer);

// Get one generate geodatabase parameter overrides.
GenerateGeodatabaseParameters streetsGdbParams =
geodatabaseParamOverrides[streetsParameterKey];

// Don't return attachments for this layer.
streetsGdbParams.ReturnAttachments = false;
```

Once you are happy with your changes, you can obtain a `GenerateOfflineMapJob` by calling `generateOfflineMap` on the offline map task supplying both the parameters and the overrides.

Copyright © 1995-2020 Esri. All rights reserved.

Considerations

- Advanced symbols are supported only if they are defined in the original service. Any overrides with advanced symbols will result in empty symbols in an offline map.
- Area-of-interest geometries that cross the dateline are not currently supported.
- If more than one feature layer in a map refers to the same feature service endpoint, only one feature layer will be taken offline. The other feature layers will raise an error.

Next steps

Once you have obtained the offline map, follow these steps:

1. Display and interact with the maps, layers, and data offline. Explore and edit the feature data, if necessary.
2. Update an offline map to synchronize your edits with the online services when connectivity is restored.
3. Finish using an offline map to unregister any geodatabases and release all locks on the mobile map package.

Copyright © 1995-2020 Esri. All rights reserved.

Take a map offline - preplanned

The preplanned workflow allows an organization to prepare offline maps so that they can be taken into the field, as required. The map author creates a map area defined by a specific geographical area of work. This process generates the associated map, tile package, and geodatabase files and stores them in a mobile map package. The offline map resides in this mobile map package, in ArcGIS Online or ArcGIS Enterprise, until the field worker downloads it. As soon as the mobile map package is downloaded, the field worker can disconnect their device and work with the map offline.

The main advantages of this workflow are as follows:

- The map author can organize and define very specific map areas for the field workers.
- The user can obtain the preplanned map area more quickly because the mobile map package has been generated, in advance, by the author. In contrast, during the on-demand workflow, the user has to both generate and download the mobile map package to their device.
- Many users can download exactly the same mobile map package.

1. Create preplanned map areas
2. Identify a map area
3. Create the parameters
4. Create the job
5. Run the job

Create preplanned map areas

The preplanned workflow allows you to create a mobile map package, in advance, so that it can be downloaded by field workers when they need it. To do this, create a map area within a web map and generate the mobile map package for the map area.

Note:
To create preplanned map areas, you must use web maps created with ArcGIS or with ArcGIS Enterprise 10.6.1 or later. You must ensure that all services used in the web map are enabled for offline use.

You can create map areas manually using the ArcGIS Online or ArcGIS Enterprise user interface, or programmatically, using Python scripts or the ArcGIS REST API, as follows:

ArcGIS Online or ArcGIS Enterprise

Manually create, edit, and manage map areas as follows:

- Take web maps offline using ArcGIS Online
- Take web maps offline using ArcGIS Enterprise

Note:
If a map area was created with a version earlier than 10.6.1, you can take it offline. If you want to edit the map area, you must re-create it with ArcGIS Portal 10.6.1 or later.

Copyright © 1995-2020 Esri. All rights reserved.

Python scripts

Use the ArcGIS API for Python to write python scripts to create map areas and generate their associated data packages. This is detailed in the managing map areas Python topic along with other useful map area management functions such as the following:

- Update a map area and any of its packages that have been deleted or contain newer data.
- Delete an individual data package.
- Delete a map area along with its associated packages.
- Inspect any of the map area's packages.
- List the web map's current map areas.

Note:
It is recommended that you use ArcGIS API for Python 1.7 or later.

ArcGIS Rest API

Use the ArcGIS REST API if you need even more control when creating your preplanned map areas. There are two stages to this process:

- To create a map area, you must run the Create Map Area task, providing a bookmark or geographical extent as the area of interest. Upon completion, the task will generate a new portal item of type `MapArea`. This task requires that the web map is enabled for offline use. It is only available to the owner of the web map and organization administrators.
- To create the data defined by the map area, you must run the Setup Map Area task, providing the map area portal item ID. This will create a set of data files such as tile packages, vector tile packages, and SQLite geodatabases. For good data management, it is strongly recommended that you also provide a folder to organize these files. This task is only available to the map area portal item owner and organization administrators.

Note:
You must use ArcGIS REST API 10.6.1 or later.

Define updates

In addition to creating the map area, you can also define how users receive updates to the geodatabases in the offline map. Updates can happen by synchronizing the offline map's geodatabases with their online services or by applying scheduled update files to the geodatabases (read-only).

- Synchronization allows you to download updates from the online service and upload edits from the offline map, once connectivity is restored. You can configure this synchronization direction to determine in which direction edits are synchronized—download only, upload only, or bi-directional.
- Apply scheduled updates allows you to schedule the regular creation of a file containing updates made to the feature service. The ArcGIS Runtine API can download the files it requires and apply them to the offline map's geodatabases. These files are relatively small so they are quick to download and fast to apply. This is a scalable solution for receiving updates to read-only geodatabases. This option is useful in cases in which you need to provide many clients with an up-to-date read-only copy of the feature data. You can set the scheduled update parameters using the Setup Map Area task in the ArcGIS REST API. Applying these updates will be described in Update an offline map.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
Scheduled updates are supported with feature services created using ArcGIS Online or ArcGIS Enterprise 10.7.1 or later. Therefore, you must use ArcGIS REST API 10.7.1 or later.

Once you have created the preplanned map areas, they will be ready to download to your device. The following section describes how you build a Runtime app to download the map area.

Identify a map area

As discussed above, each web map can contain one or more preplanned map areas. Your app needs to determine which map area should be downloaded. You can do this by allowing the field worker to select a map area from a map view or from a list of map areas. Alternatively, the app logic could select a map area for the field worker.

Follow these steps to retrieve all of the map areas:

1. Instantiate the offline map task by passing either a map (created from a web map) or a web map's portal item to the `OfflineMapTask` constructor.

```
// Get a web map item using its ID.
PortalItem webmapItem = await PortalItem.CreateAsync(portal,
"acc027394bc84c2fb04d1ed317aac674");

// Create a map from the web map item.
Map onlineMap = new Map(webmapItem);

// Create an OfflineMapTask from the map ...
takeMapOfflineTask = await OfflineMapTask.CreateAsync(onlineMap);
// ... or a web map portal item.
takeMapOfflineTask = await OfflineMapTask.CreateAsync(webmapItem);
```

2. Retrieve a list of the preplanned map areas from the web map using the `GetPreplannedMapAreasAsync` method on the `OfflineMapTask`.
3. Load the preplanned map area; `PreplannedMapArea`.
4. Display the map area titles and thumbnails in a list, or show the map area extents or just select a map area by name.

```
// Get a list of the available preplanned map areas.
IReadOnlyList<PreplannedMapArea> preplannedMapAreaList = await
takeMapOfflineTask.GetPreplannedMapAreasAsync();

// Loop through all preplanned map areas.
foreach (PreplannedMapArea mapArea in preplannedMapAreaList)
{
    // Load the preplanned map area so property values can be read.
    await mapArea.LoadAsync();

    // Get the area of interest (geometry) for this area.
    Geometry aoi = mapArea.AreaOfInterest;

    // Get the portal item for this area, read the title and thumbnail image.
    PortalItem preplannedMapItem = mapArea.PortalItem;
    string mapTitle = preplannedMapItem.Title;
    RuntimeImage areaThumbnail = preplannedMapItem.Thumbnail;
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Create the parameters

To take a map offline you can provide a set of `DownloadPreplannedOfflineMapParameters`. To obtain a set of default parameters pass in the map area to the `CreateDefaultDownloadPreplannedOfflineMapParametersAsync` method on the `OfflineMapTask`. The value of these default parameters will match the advanced offline settings configured by the web map author (discussed in Take web maps offline.

```
// Get the default parameters (provide a map area).
DownloadPreplannedOfflineMapParameters defaultDownloadParams =
    await
takeMapOfflineTask.CreateDefaultDownloadPreplannedOfflineMapParametersAsync(downtownMapArea);

// Update some of the parameter values.
defaultDownloadParams.ContinueOnErrors = false;
defaultDownloadParams.IncludeBasemap = true;
defaultDownloadParams.ReferenceBasemapDirectory = dataFolder;
```

For more information see Advanced parameters.

Create the job

Create the download preplanned map area job by calling the `DownloadPreplannedOfflineMap` method on the `OfflineMapTask`. Pass in the `PreplannedMapArea` and a download directory on the device. If this download directory already exists it must be empty. If the directory doesn't exist, it will be created by the job.

```
DownloadPreplannedOfflineMapJob preplannedMapJob =
takeMapOfflineTask.DownloadPreplannedOfflineMap(downtownMapArea, pathToOutputPackage);
```

If, however, you have created parameters to control the map content you must create the `DownloadPreplannedOfflineMapJob` using the `DownloadPreplannedOfflineMap` method on the `OfflineMapTask` and provide both the `DownloadPreplannedOfflineMapParameters` and the download directory.

```
preplannedMapJob = takeMapOfflineTask.DownloadPreplannedOfflineMap(parameters,
pathToOutputPackage);
```

Run the job

To download the preplanned map area to your device start the `DownloadPreplannedOfflineMapJob`. Upon completion the job will return an instance of the `DownloadPreplannedOfflineMapResult`. If at least one table or layer failed to be taken offline the `HasErrors` property will be `true`. In this case you should examine the `layerErrors` and `tableErrors` dictionaries to identify the problem.

Remember that if the job's `ContinueOnErrors` parameter has been set to false then the job will be terminated immediately if a single layer or table fails to be taken offline.

If you want to display the map immediately then use the `DownloadPreplannedOfflineMapResult.OfflineMap`.

```
// Generate the offline map and download it.
DownloadPreplannedOfflineMapResult preplannedMapResult = await preplannedMapJob.GetResultAsync();
```

Copyright © 1995-2020 Esri. All rights reserved.

```
if (!preplannedMapResult.HasErrors)
{
    // Job completed successfully and all content was generated.
    Debug.WriteLine("Map " +
        preplannedMapResult.MobileMapPackage.Item.Title +
        " was saved to " +
        preplannedMapResult.MobileMapPackage.Path);

    // Show the offline map in a MapView.
    MyMapView.Map = preplannedMapResult.OfflineMap;
}
else
{
    // Job is finished but one or more layers or tables had errors.
    if (preplannedMapResult.LayerErrors.Count > 0)
    {
        // Show layer errors.
        foreach (var layerError in preplannedMapResult.LayerErrors)
        {
            Debug.WriteLine("Error occurred when taking " +
                layerError.Key.Name +
                " offline. Error : " +
                layerError.Value.Message);
        }
    }
    if (preplannedMapResult.TableErrors.Count > 0)
    {
        // Show table errors.
        foreach (var tableError in preplannedMapResult.TableErrors)
        {
            Debug.WriteLine("Error occurred when taking " +
                tableError.Key.TableName +
                " offline. Error : " +
                tableError.Value.Message);
        }
    }
}
```

Note:

Note that the job will behave the same whether it was created with or without the `DownloadPreplannedOfflineMapParameters`.

For more details on how to work with jobs, see the Tasks and jobs topic.

Offline maps created by the preplanned workflow are stored in an unpacked mobile map package. When your app goes into the field you will need to open the map directly from the mobile map package `downloadDirectory` stored on your device. See Open an offline map for more information.

Advanced parameters

Once you have the default parameters you can customize them as required. For example:

- Continue if a table or layer fails?
  The `DownloadPreplannedOfflineMapJob` takes a map's tables and layers offline. By default this job will continue to do this even if one table of layer has failed. Failures could be due to an intermittent network connection, loss of the service or an unsupported layer type. The approach ensures that you can take a map offline but you may have some data missing.

Copyright © 1995-2020 Esri. All rights reserved.

To ensure that the offline map contains all of its layers and tables you can request that the job is stopped if any layer or table fails to download. To do this set the `ContinueOnErrors` property to `false`.

- Which basemap should the map use?
  The author of the web map can define whether the map uses:
  - The basemap defined by the web map
    This is the default situation and ensures that a tile package is downloaded as part of the mobile map package.
  - A tile package that is already on the device
    The tile package must be downloaded or copied onto the device and can be located using an actual file path on the device or a path that is relative to the mobile map package. You must ensure that the tile package covers the areas required by your map area. The benefits of this option are that the mobile map package file will be smaller, the download time may be faster, and you can use the tile package in many different maps and apps.

    To use the tile package on your device you must set the `referenceBasemapDirectory` to the directory which contains the tile package. It is recommended that you confirm that the tile package file, `referenceBasemapFilename`, exists on the device before running the `AGSDownloadPreplannedOfflineMapJob`. This job will add the tile package, as a basemap, to the offline map.

```
// Use the ReferenceBasemapFilename value in the default parameters to create a path to the
location of the expected tile package.
string path = System.IO.Path.Combine(dataFolder,
defaultDownloadParams.ReferenceBasemapFilename);

// If the package exists, set the ReferenceBasemapDirectory so
DownloadPreplannedOfflineMapJob can use the local tiles as the map's basemap.
if (System.IO.File.Exists(path))
{
    defaultDownloadParams.ReferenceBasemapDirectory = dataFolder;
}
```

    Note:
    If this tile package has a different spatial reference to your online map then the offline map will use the spatial of the tile package.

  - No basemap
    If you only want to take the operational layers offline you can programmatically exclude the basemap. To do this set the `includeBasemap` property to false. In this case the `DownloadPreplannedOfflineMapJob` will not download layers included as part of the map's `basemap`. This task will not use the local tile package, even if you have specified one.

- Overwrite the geodatabase update mode?
  If the web map author has specified that scheduled updates will be generated for all feature services in the map (discussed in Define updates, then the `updateMode` on the `DownloadPreplannedOfflineMapParameters` will be set to `PreplannedUpdateModeDownloadScheduledUpdates`. This means that the geodatabases will be read-only and can only receive updates via the scheduled update files.

  If you want to edit these geodatabases and adopt the synchronization approach you can override this `updateMode` by setting it to `PreplannedUpdateModeSyncWithFeatureServices`. From this point you will not be able to update the geodatabase using the scheduled update files.

Copyright © 1995-2020 Esri. All rights reserved.

Note:

Instead of obtaining the default set of `DownloadPreplannedOfflineMapParameters` you could construct them manually. The author of the web map can recommended how data is downloaded to offline devices, and subsequently synchronized, by adjusting the map's advanced offline settings (discussed in Take web maps offline). You can access these advanced settings from the map's `OfflineSettings`. They include:

- working with feature attachments
- synchronizing edits
- downloading a basemap or using one which is already on the device

Next steps

Once you have obtained the offline map follow these steps:

1. Display and interact with the maps, layers and data offline. Explore and edit the feature data, if necessary.
2. Update an offline map by synchronizing your edits with the online services, when connectivity is restored.
3. Finish using an offline map by unregistering any geodatabases and fully releasing all locks on the mobile map package.

Copyright © 1995-2020 Esri. All rights reserved.

Take a map offline - ArcGIS Pro

ArcGIS Pro allows you to consolidate your maps and data into a single mobile map package file (`.mmpk`) that you can take offline. Each package can contain the maps and data necessary for your users to continue working when connectivity is poor or non-existent. Each mobile map package adheres to a common map definition, allowing you to transport your maps and data across the ArcGIS platform.

With ArcGIS Pro 2.6 (or later), you can include online layers and tables in your map, such as traffic information, weather conditions, or a large company data-set. If your user's device has network connectivity they can take advantage of these online layers and tables. If the connection is unavailable, users can continue to work with the maps and data that are stored locally in the mobile map package.

Create an offline map

You have two options to create a mobile map package with ArcGIS Pro:

- Use the Create Mobile Map Package tool.
  - Locate the ArcGIS Pro tool called Create Mobile Map Package in the Data Management Toolbox. You can use this to create a mobile map package containing any number of maps.
  - If needed, set an expiration date on the mobile map package. This is available with ArcGIS Pro 2.4 (or later) with the ArcGIS Publisher extension license.
  - If you want include online layers and tables in the maps, you need to check on the "Reference online content". This is available with ArcGIS Pro 2.6 (or later).
  - Save the package to ArcGIS Online, ArcGIS Enterprise, or to a folder on your device.
- Use the Package option on the Share Tab.
  - In ArcGIS Pro go to the **Share** tab, under **Package** and select **Mobile Map**. This will create a mobile map package from the active map. For more information, see Share a mobile map package.
  - You can save the package to ArcGIS Online, ArcGIS Enterprise, or to a folder on your device.

You can download the mobile map package from ArcGIS Online or ArcGIS Enterprise directly onto your device using the ArcGIS Runtime API. Alternatively, you can copy the package directly to your device.

Note:
If you'd like ready-to-use and regularly updated basemaps, locators, or network datasets you can license StreetMap Premium data in mobile map package format. For details, see Add StreetMap Premium data.

Next steps

When the mobile map package is on your device, you can:

1. Open the offline map. Ensure that the mobile map package has not expired and load it into your application. If any of the loaded maps contain online layers or tables, you may need to handle authentication.
2. Display and interact with the maps, layers, and data, calculate routes using the package's transportation network files, and search for addresses using the package's locator files, if present.
3. Finish using the offline map by ensuring that all locks are fully released from the mobile map package.

Copyright © 1995-2020 Esri. All rights reserved.

Open an offline map

You can open any map stored in any mobile map package, that was created by any of the following workflows:

- Take a map offline - preplanned—Provides a mobile map package directory.
- Take a map offline - on-demand—Provides a mobile map package directory.
- Take a map offline - ArcGIS Pro—Provides a mobile map package file (`.mmpk`).

If you created the mobile map package with ArcGIS Pro, you may need to consider whether the mobile map package content has expired. If not, proceed to open the offline map.

Handle expired mobile map packages

Using ArcGIS Pro 2.4 (or later), you can control the lifetime of your maps by setting an expiration date on your mobile map package. This will allow you to do the following:

- Provide data to third-party contractors that will expire at the end of a contract.
- Indicate to your users that the data has become obsolete.

When you use the ArcGIS Pro tool to create the mobile map package, you can set an expiration date and time, an expiration message, and access options if the package has expired.

Note:
These mobile map package expiration properties are available if ArcGIS Pro is licensed with the ArcGIS Publisher extension.

As an ArcGIS Runtime developer, there are three possible outcomes when you try to load a mobile map package:

1. You have full access to the mobile map package. This situation occurs if the mobile map package has not expired or if it has been created without any expiration properties.
2. You have full access to the mobile map package even though the maps and their data have expired. You will be warned that the mobile map package is out of date.
3. You cannot access the mobile map package because it has expired. In this case, the mobile map package will fail to load and you will not be able to access its maps and data.

These three outcomes are managed by the mobile map package load process `MobileMapPackage.LoadAsync()`. No extra workflow or code paths are required to accommodate this expiration.

If you want to provide your users with extra information, you can retrieve the expiration details from the `Expiration` property on the `MobileMapPackage`. For example, you can display how many days remain before the mobile map package expires. To do this, perform the following steps:

1. Load the mobile map package.
2. Read the expiration property on the mobile map package.
3. Confirm that the mobile map package has not expired.
4. Get the expiration date.
5. Calculate the number of days between today and the expiration date. Report the number of days remaining.

```
// Get expiration details from the package.
Expiration expirationDetails = mobileMapPackage.Expiration;
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create a string builder to report expiration status to the user.
System.Text.StringBuilder messageBuilder = new System.Text.StringBuilder();

// See if the package is past the expiration date.
if (expirationDetails.IsExpired)
{
    messageBuilder.AppendLine("The mobile map package has expired:");

    // See if the package can still be used despite expiration.
    switch (expirationDetails.Type)
    {
        case ExpirationType.AllowExpiredAccess:
            messageBuilder.AppendLine("You can continue to use this package.");
            messageBuilder.AppendLine("Consider obtaining a current, updated version.");
            break;
        case ExpirationType.PreventExpiredAccess:
            messageBuilder.AppendLine("This package can no longer be used.");
            ShowMessage(messageBuilder.ToString());
            return;
    }
}
else
{
    // If package has not expired, let the user know how many days remain.
    System.DateTimeOffset expirationDate = expirationDetails.DateTime;
    double daysLeft = expirationDate.Subtract(System.DateTime.Today).TotalDays;
    messageBuilder.AppendLine(string.Format("The package will expire in {0} days.",
daysLeft.ToString("F2")));

    // Warn if expiration will prevent the package from being used.
    if(expirationDetails.Type == ExpirationType.PreventExpiredAccess)
    {
        messageBuilder.AppendLine("You will not be able to use this mobile map package after
that time.");
    }
}
```

Open the offline map

You can open the map directly from a mobile map package file (created by ArcGIS Pro) or from a mobile map package directory (created by the preplanned or on-demand workflows). To create the mobile map package object, pass the mobile map package file path or directory to the `MobileMapPackage` constructor. Load the mobile map package and access its collection of maps. Passing the map to the map view will initiate loading the map, its layers, and data.

```
// Create a mobile map package from an unpacked map package folder.
MobileMapPackage offlineMapPackage = await MobileMapPackage.OpenAsync(pathToUnpackedPackage);

// Set the title from the package metadata to the UI
Title.Text = offlineMapPackage.Item.Title;

// Get the map from the package and set it to the MapView
var map = offlineMapPackage.Maps.First();
MyMapView.Map = map;
```

Note:

If the mobile map package was created with ArcGIS Pro 2.6 (or later) it may contain online layers or tables. If so, then your app may need to handle authentication before the layers and tables are loaded.

Copyright © 1995-2020 Esri. All rights reserved.

After loading the map, its metadata is accessible using the `Map.Item` property. In some situations, you may want to access the metadata from the mobile map package without loading the map itself. You can access this from the `MobileMapPackage.Item` property.

See Display a map for more details on how to access and display maps stored in mobile map packages.

Next steps

After you have loaded and opened the mobile map package, you can continue to work with the maps and follow these steps:

1. Display and interact with the maps, layers, and data offline. Explore and edit the feature data, if necessary.
2. Update an offline map by synchronizing your edits with the online services when connectivity is restored.
3. Finish using an offline map to unregister any geodatabases and fully release all locks on the mobile map package.

Copyright © 1995-2020 Esri. All rights reserved.

Update an offline map

Maps that contain feature layers store their feature data in .geodatabase files, within a mobile map package, when it is taken offline. This allows users to continue to work with features, display their location, and analyze their spatial and non-spatial attributes, even when connectivity is lost. The extent to which these features can be updated (by the user or from an online service) depends on how the mobile map package was created. The following table summarizes the types of feature updates that are supported for each of the different offline workflows.

Supported feature updates

	ArcGIS Pro	**on-demand**	**preplanned**
Add, edit and create features in the local mobile map package. See edit features.	No	Yes	Yes
Synchronize the features in the mobile map package with their online feature service. The online feature service must support synchronization. See sync offline edits.	No	Yes	Yes
Apply scheduled updates from the online feature service to the mobile map package. Feature layers updated in this way are read-only. See apply scheduled updates.	No	No	Yes
Add, edit and create features in the online feature layer. Connectivity must be present and authentication may be required. See edit features.	Yes	No	No

If the mobile map package was created with ArcGIS Pro then the features in the geodatabase can not be edited (read-only) and cannot be updated from the online service. You can only provide updates to the features by creating a brand new mobile map package, as described in Take a map offline - ArcGIS Pro.

If the mobile map package was created using the on-demand or preplanned workflows, and synchronization was enabled on the online feature service, the features can be edited and synchronized back with the online feature service when connectivity is restored. You can synchronize changes with a single geodatabase, or with all geodatabases in the map, as described in Sync offline edits..

If the mobile map package was created using the preplanned workflow, the map author can schedule regular updates (delta files) to be generated for the individual map areas. This is described in define updates. Whenever connectivity is restored, the app can check for these scheduled updates, download any delta files and update the map's geodatabases. You can only apply scheduled updates to read-only data. The process is described in the following section:

Apply scheduled updates

1. Create the `OfflineMapSyncTask` from the online map used in the preplanned workflow.

```
OfflineMapSyncTask offlineMapSyncTask = await OfflineMapSyncTask.CreateAsync(MyMapView.Map);
```

2. Retrieve the task's `OfflineMapUpdateCapabilities` and check that it supports scheduled updates.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Get the update capabilities for the task.
OfflineMapUpdateCapabilities updateCapabilities = offlineMapSyncTask.UpdateCapabilities;
if (updateCapabilities.SupportsScheduledUpdatesForFeatures)
{
    Console.WriteLine("Scheduled updates are supported.");
}
else if (updateCapabilities.SupportsSyncWithFeatureServices)
{
    Console.WriteLine("Synchronization with feature services is supported.");
}
```

3. Determine whether there are any updates available by calling the task's `checkForUpdatesWithCompletion` method.
   - If the map `supportsScheduledUpdatesForFeatures`, you can examine the `downloadAvailability` on the `OfflineMapUpdatesInfo` object. If there are downloads available, you can check to see how large the download is by using the `scheduledUpdatesDownloadSize`. You can use this information to determine whether to download updates immediately—for example, based on available disk space or network availability.
   - If the map `supportsSyncWithFeatureServices`, you can examine the `uploadAvailability` to see whether the offline map has any edits to upload. If so, go to step 4. It is not possible to determine whether there are online changes to download without performing a full feature synchronization.

```
// Are there any updates available?
OfflineMapUpdatesInfo updateInfo = await offlineMapSyncTask.CheckForUpdatesAsync();
if(updateInfo == null)
{
    Console.WriteLine("There are no updates available.");
    return;
}

// Are there any downloads available?
switch (updateInfo.DownloadAvailability)
{
    case OfflineUpdateAvailability.Available:
        Console.WriteLine("Online edits are available for download.");
        Console.WriteLine(string.Format("Download size is {0} bytes.",
updateInfo.ScheduledUpdatesDownloadSize));
        break;
    case OfflineUpdateAvailability.None:
        Console.WriteLine("No online edits are available for download.");
        break;
    case OfflineUpdateAvailability.Indeterminate:
        Console.WriteLine("Unable to determine if edits are available for download.");
        break;
}

// Are there uploads available?
switch (updateInfo.UploadAvailability)
{
    case OfflineUpdateAvailability.Available:
        Console.WriteLine("Local edits are available to upload.");
        break;
    case OfflineUpdateAvailability.None:
        Console.WriteLine("No local edits are available to upload.");
        break;
    case OfflineUpdateAvailability.Indeterminate:
        Console.WriteLine("Unable to determine if edits are available to upload.");
        break;
}
```

Copyright © 1995-2020 Esri. All rights reserved.

4. Obtain the default offline map sync parameters using the method. Override any of these parameters, as required. For example, if resources are limited, you may want to overwrite the sync direction to just `Upload` local edits to the online service.
5. Create the offline map sync job, `offlineMapSyncJob`, using the offline map sync parameters and run it.
6. Upon completion, check the `OfflineMapSyncResult` for any errors.
7. If `supportsScheduledUpdatesForFeatures`, determine whether the mobile map file needs to be reopened by examining the `mobileMapPackageReopenRequired` property on the `OfflineMapUpdatesInfo` object. If so, close the offline map before reopening it.

```
// Get the default parameters for the offline map sync task.
OfflineMapSyncParameters syncParams = await
offlineMapSyncTask.CreateDefaultOfflineMapSyncParametersAsync();
// (Change individual parameter values as needed).

// Create the offline map sync job with the parameters.
OfflineMapSyncJob syncJob = offlineMapSyncTask.SyncOfflineMap(syncParams);

// Handle the progress changed event to show the percent complete.
syncJob.ProgressChanged += (s, e) =>
{
    Console.WriteLine(string.Format("Offline sync job is {0}% complete.", syncJob.Progress));
};

// Handle the job changed event to react to job success or failure.
syncJob.JobChanged += async(s, e) =>
{
    if(syncJob.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Succeeded)
    {
        OfflineMapSyncResult syncResult = await syncJob.GetResultAsync();
        if (syncResult.HasErrors)
        {
            // Report errors.
            return;
        }

        if (syncResult.IsMobileMapPackageReopenRequired)
        {
            // Close and reopen the mobile map package.
        }
    }
    else if(syncJob.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Failed)
    {
        // Report errors.
    }
};

// Start the sync job.
syncJob.Start();
```

Next

When you have finished working with the mobile map package, ensure that you release its maps, layers, and geodatabases before closing or terminating the application. See Finish using an offline map for more details.

Copyright © 1995-2020 Esri. All rights reserved.

Finish using an offline map

When you have finished using a mobile map package, it is recommended that you remove the mobile map package, to free up disk space on your device, and remove replicas on online feature services. You should do the following:

- Unregister any geodatabases that are no longer required for synchronization.
- Close all mobile map packages that are no longer required.

Unregister a geodatabase

Geodatabases are registered with an online sync-enabled feature service to ensure that any of their feature updates can be synchronized with the feature service. A geodatabase is registered automatically when you do the following:

- Take a map offline using the preplanned workflow.
- Take a map offline using the on-demand workflow.
- Take a feature layer offline using the geodatabase sync task. See Take a layer offline.

You can also register it manually. If you do not intend to edit and synchronize your changes or you have stopped work with this geodatabase, you should unregister the geodatabase from the online feature service. Unregistering it removes the geodatabase's replica ID from the service so it can no longer be synchronized. You need to do this when your device is online and before you delete the mobile map package.

```
// Create a new GeodatabaseSyncTask using the online feature service.
GeodatabaseSyncTask gdbSyncTask = await GeodatabaseSyncTask.CreateAsync(featureServiceUri);

try
{
    // Unregister the local geodatabase that was originally generated from the service.
    await gdbSyncTask.UnregisterGeodatabaseAsync(localGeodatabase);
}
catch(Exception exp)
{
    ShowMessage("Error while unregistering database '" + localGeodatabase.Path + "'" +
exp.Message);
}
```

You can perform this operation later if your device cannot be online when you delete the mobile map package. Obtain the Sync ID from the geodatabase, store this, and use this to unregister the geodatabase when your device is online.

```
// Store the sync ID of the local geodatabase.
Guid syncId = localGeodatabase.SyncId;

// Create a new GeodatabaseSyncTask using the online feature service.
GeodatabaseSyncTask gdbSyncTask = await GeodatabaseSyncTask.CreateAsync(featureServiceUri);

try
{
    // Unregister the local geodatabase using the sync ID.
    await gdbSyncTask.UnregisterGeodatabaseAsync(syncId);
}
catch (Exception exp)
{
    ShowMessage("Error while unregistering database '" + localGeodatabase.Path + "'" +
exp.Message);
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Close the mobile map package

When a mobile map package is loaded and used by a Runtime app, the operating system will lock the underlying mobile map package file or unpacked package files until the memory is freed or garbage collection runs (depending on the platform). These locks prevent files from being inadvertently moved or deleted. It is strongly recommended that you close a mobile map package if you need to move, delete, or reopen it while the application is running. Reopening a mobile map package may be required after applying scheduled updates. Close the mobile map package, as follows:

1. Stop using the mobile map package. Remove the map from the map view. Remove all references to the mobile map package including maps, layers, geodatabases, networks, locators, and so on.
2. Call the mobile map package's `close` method to free its locks. If active references to mobile package data exist, the `close` method will still close the package and subsequent rendering and data access methods will fail.
3. The underlying mobile map package file or directory can be moved or deleted and the app can be terminated.

```
_mobileMapPkg.Close();
```

Next

When you have released all references to the contents of the mobile map package and closed it, you can close your application and delete or move the mobile map package, if required.

Copyright © 1995-2020 Esri. All rights reserved.

Work with offline layers

Layers of data can be sourced from either online services such as ArcGIS Online, WFS, or WMS, or offline sources such as mobile geodatabases, shapefiles, GeoPackages, tile packages, and raster datasets. Using ArcGIS Pro and ArcMap, you can create these GIS files and packages and deliver them via email, FTP, the cloud, networks, thumb drives, and so on. The recipient just copies (or sideloads) the files and directories onto their device so that the ArcGIS Runtime apps can work offline without a network connection.

ArcGIS Runtime supports the following offline layers:

- Tiled layer data stored in tile packages (`.tpk` or `.tpkx`).
- Vector tiled layer data stored in vector tile packages (`.vtpk`).
- Feature layer data stored in the following formats:
  - ArcGIS features held in a mobile geodatabases (`.geodatabase`).
  - Features stored in a GeoPackages provisioned by the OGC file format (`.gpkg`).
  - Features stored in the shapefile format provided by the `.shp` file and its associated files (`.dbf`, `.shx`, and so on).
- Raster layer data stored in a raster dataset file or in a GeoPackage (`.gpkg`).

ArcGIS Pro and ArcMap provide a range of tools to create these GIS files and packages. The following sections demonstrate how you prepare these data files, add them to your ArcGIS Runtime app, and create layers to display in your map.

Tiled layer

Tiled layers are typically used to display pregenerated tiled data as basemaps. These give geographical context to operational layers and graphics in your map. You can take a portion of tiled data offline and store it within a single tile package (`.tpk` or `.tpkx`) file. To do this, you need to specify area of interest, the tiling scheme, the levels of detail, and the tile format. You can do so by using any of the following tools:

- Run the ArcGIS Pro python tool, Create Map Tile Package, to create a tile package file.
- In ArcMap, choose **File** > **Share As** > **Tile Package** to create a tile package file, as described in the How to create a tile package help topic.
- In ArcMap, choose **Share as > ArcGIS Runtime Content** to export the map's basemap layer to a tile package file (`.tpk`) that is output within the ArcGIS Runtime Content folder. This is described in the Creating ArcGIS Runtime content help topic, which is available with ArcGIS 10.2.1 for Desktop or later.

Review this document for more information about tile packages.

Note:

When you create a tile package, it must have the same spatial reference as the map in which it will be displayed.

To create a tiled layer from a tile package file, instantiate an `ArcGISTiledLayer` object with the path to the tile package file on the device.

```
// Path to the local package (.tpk file).
string myDocumentsFolder = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string tileCachePath = Path.Combine(myDocumentsFolder, "StreetMapTiles.tpk");

// Create a tile cache from the local data.
```

Copyright © 1995-2020 Esri. All rights reserved.

```
TileCache cache = new TileCache(tileCachePath);

// Use the tile cache to create an ArcGISTiledLayer.
ArcGISTiledLayer tiledLayer = new ArcGISTiledLayer(cache);

// Display the tiled layer as a basemap.
MyMapView.Map = new Map(new Basemap(tiledLayer));
```

Vector tiled layer

Vector tiled layers contain vector representations of data across a range of scales. Unlike raster tiles, they can adapt to the resolution of their display device as you zoom in and out. You can take an area of vector tiled data offline by exporting the vector tiles to a vector tile package file (`.vtpk`) using the Create Vector Tile Package tool in ArcGIS Pro. Be aware that ArcGIS Pro does not support custom styles and only packages the default styles into the vector tile package.

Note:
When you create a vector tile package, it must be in the same spatial reference as the map in which it will be displayed.

To create a vector tiled layer from the vector tile package (`.vtpk`) instantiate an `ArcGISVectorTiledLayer` object with the vector tile package's file URL. The default style will be loaded directly from the vector tile package.

```
// Path to the local vector tile package (.vtpk file).
string myDocumentsFolder = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string tileCachePath = System.IO.Path.Combine(myDocumentsFolder, "PencilMapTiles.vtpk");

// Create a vector tile cache from the local data.
VectorTileCache cache = new VectorTileCache(tileCachePath);

// Use the tile cache to create an ArcGISVectorTiledLayer.
ArcGISVectorTiledLayer tiledLayer = new ArcGISVectorTiledLayer(cache);

// Display the vector tiled layer as a basemap.
MyMapView.Map = new Map(new Basemap(tiledLayer));
```

Feature layers

Feature layers allow you to display, select, and query individual features and their attributes. If supported, you can also edit features, their attributes, and their attachments. The desktop pattern allows you to work with features offline using cached features stored in an offline file, such as a geodatabase file (`.geodatabase`), a GeoPackage file (`.gpkg`), or a shapefile (`.shp`).

ArcGIS features

To generate a mobile geodatabase, use the **Share as > ArcGIS Runtime Content** menu item as described in the ArcMap help topic Creating ArcGIS Runtime content. This is available with ArcGIS for Desktop 10.2.1 or later.

To create an offline feature layer from a mobile geodatabase (`.geodatabase`), complete the following steps:

1. Instantiate the `Geodatabase` object with a path to the geodatabase file.
2. Load the geodatabase and instantiate a `FeatureTable` from one of the geodatabase's feature tables.
3. Create an `FeatureLayer` from the `FeatureTable` and add it as an operational layer to the map.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Get the path to a local geodatabase.
string geodatabasePath = Path.Combine(Environment.ExpandEnvironmentVariables("%TEMP%"),
"recreation.geodatabase");

// Open the geodatabase by providing the path.
Geodatabase localGdb = await Geodatabase.OpenAsync(geodatabasePath);

// Access one of the feature tables in the geodatabase using its name.
GeodatabaseFeatureTable trailheadsTable = localGdb.GeodatabaseFeatureTable("Trailheads");

// Create a feature layer from the table and add it to the map's operational layers.
FeatureLayer trailheadsLayer = new FeatureLayer(trailheadsTable);
MyMapView.Map.OperationalLayers.Add(trailheadsLayer);
```

Geopackage

GeoPackage is an open, standards-based, platform-independent, portable, self-describing, compact format for transferring geospatial information. It uses a single SQLite file (`.gpkg`) that conforms to the OGC GeoPackage standard. You can create a GeoPackage file (`.gpkg`) from your own data using the Create SQLite Database tool in ArcGIS Pro.

To display features stored in a GeoPackage file, you must do the following:

1. Instantiate the `GeoPackage` with the `.gpkg` file path.
2. Load the GeoPackage and then examine its list of `GeoPackageFeatureTables`.
3. Create a `FeatureLayer` from one of the `GeoPackageFeatureTables` and add it as an operational layer to the map.

```
// Get the path to a local geopackage.
string geopackagePath = Path.Combine(Environment.ExpandEnvironmentVariables("%TEMP%"),
"trails.gpkg");

// Open the geopackage by providing the path.
GeoPackage trailsGpkg = await GeoPackage.OpenAsync(geopackagePath);

// Iterate feature tables in the package to find one by name.
foreach (GeoPackageFeatureTable table in trailsGpkg.GeoPackageFeatureTables)
{
    if (table.TableName.ToLower() == "trailheads")
    {
        // Create a feature layer from the table and add it to the map's operational layers.
        FeatureLayer trailheadsLayer = new FeatureLayer(table);
        MyMapView.Map.OperationalLayers.Add(trailheadsLayer);
    }
}
```

Shapefiles

To create a feature layer from a shapefile (`.shp`), do the following:

1. Instantiate the `ShapefileFeatureTable` with the path to the shapefile. This path must point to the `.shp` file and the associated `.shx` and `.dbf` files must be present at the same location.
2. Create an `FeatureLayer` from the `ShapefileFeatureTable` and add it as an operational layer to the map.

```
// Get the path to a local shapefile.
string shapefilePath = Path.Combine(Environment.ExpandEnvironmentVariables("%TEMP%"),
"trails.shp");
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create a shapefile feature table using the path.
ShapefileFeatureTable trailsFeatureTable = await ShapefileFeatureTable.OpenAsync(shapefilePath);

// Create a feature layer from the table and add it to the map's operational layers.
FeatureLayer trailsLayer = new FeatureLayer(trailsFeatureTable);
MyMapView.Map.OperationalLayers.Add(trailsLayer);
```

Raster layer

Raster data consists of a matrix of cells in which each individual cell contains a value representing information. For example, satellite or aerial images and photographs for visualizing an area. You can define renderers to display the raster data. This SDK supports several raster formats. To work offline, just copy these onto your device and add the raster dataset to your app using the `RasterLayer` class. See Add raster data for more information.

Copyright © 1995-2020 Esri. All rights reserved.

Take a layer offline

Consider taking individual layers offline if you want to construct the map yourself and provide users with up-to-date information from ArcGIS Online or ArcGIS Enterprise. This topic describes how to take the following layers offline:

- Feature and annotation layers—Download a specific geographical area from a sync-enabled feature service into a mobile geodatabase. Edits can be synchronized with other users when connectivity is restored. This synchronization process allows the app to maintain an up-to-date view of the feature data.
- Tiled layers
  - Tiled data—Tiled ArcGIS map services, tiled ArcGIS image services, or tiled basemaps in ArcGIS Online
  - Vector tiled data—ArcGIS vector tile services or vector basemaps in ArcGIS Online

You can export, download, and store these caches locally as packages and access them while your device is offline. You cannot edit this data, and if you require updates to the data, you must export and download them again.

Feature and annotation layers

You can take features (including annotation) offline, from a feature service, by downloading them to geodatabase feature tables within a mobile geodatabase. To do this, the feature service must be hosted on ArcGIS Enterprise 10.2.2 or later for features or from a service hosted in ArcGIS Online, provided that the feature service has been sync enabled (allow disconnected editing with synchronization).

Note:
Annotation layers can only be taken offline from a feature service hosted on ArcGIS Enterprise 10.7.1 or later. Take annotation offline using a geodatabase sync task as described below. Annotation is read-only.

You can use an existing feature service or create features services for your own data. To enable sync for a feature service, do the following:

- ArcGIS Online—Edit the feature service item and check the **Sync** check box.
- ArcGIS Enterprise—See Prepare data for offline use in the ArcGIS Server documentation.

To create the mobile geodatabase, you must do the following:

1. Generate geodatabase parameters and define the area of interest, the layers, any expression filters, and so on, if required.
2. Create the geodatabase, populate it with all the relevant features, and download it to the user's device.

Generate geodatabase parameters

When you create a mobile geodatabase, you must provide a set of parameters, described below, to define exactly which data is downloaded.

- The geographical area of interest. You typically supply the area of interest as an extent (in other words, an envelope) but point, line, and polygon (including multipart) geometries are also supported. This allows you to create more detailed areas of interest. Regardless of the geometry, any features that intersect with the supplied area of interest are extracted.
- The spatial reference of the mobile geodatabase.
- Individual layers can be managed using the `GenerateGeodatabaseParameters.LayerOptions` property:
  - Determine which layers are included in the mobile geodatabase.
  - Subset the features by providing an expression that filters features by attribute values, such as `ZONE = 'COM'`.

Copyright © 1995-2020 Esri. All rights reserved.

- The synchronization model controls how edits made to the mobile geodatabase are applied back to the feature service during synchronization. The model supported is defined by the data that was used to create the sync-enabled feature service. If the data is nonversioned, the synchronization model is per layer. This is the most flexible model, allowing you to synchronize on a layer-by-layer basis, based on the layers you specify. If the data is versioned, the synchronization model is per geodatabase. This synchronizes the entire geodatabase, including all layers and tables, at once.
- Specify whether to include feature attachments in the mobile geodatabase and whether they can be uploaded during synchronization.
- Identify whether tables related to the layer are also included in the geodatabase.

You can obtain a default set of parameters (`GenerateGeodatabaseParameters`) using the `CreateDefaultGenerateGeodatabaseParametersAsync` method on the `GeodatabaseSyncTask`. If you provide the area of interest, the default parameters will be generated taking into account the capabilities supported by the ArcGIS feature service. You can update these default parameter values before creating the geodatabase.

```
// Create a new GeodatabaseSyncTask to create a local version of feature service data.
Uri featureServiceUri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/Sync/
SaveTheBaySync/FeatureServer");
GeodatabaseSyncTask gdbSyncTask = await GeodatabaseSyncTask.CreateAsync(featureServiceUri);

// Define an extent for the features to include.
Envelope dataExtent =
MyMapView.GetCurrentViewpoint(ViewpointType.BoundingGeometry).TargetGeometry as Envelope;

// Get the default parameters for generating a geodatabase.
GenerateGeodatabaseParameters generateGdbParams = await
gdbSyncTask.CreateDefaultGenerateGeodatabaseParametersAsync(dataExtent);

// Set the synchronization model to per layer.
generateGdbParams.SyncModel = SyncModel.Layer;

// Define the layers and features to include.
int marineLayerId = 0;
int birdsLayerId = 1;
string dolphinsOnlyWhereClause = "type = 11";

// Clear and re-create the layer options.
generateGdbParams.LayerOptions.Clear();
generateGdbParams.LayerOptions.Add(new GenerateLayerOption(marineLayerId,
dolphinsOnlyWhereClause));
generateGdbParams.LayerOptions.Add(new GenerateLayerOption(birdsLayerId));

// Do not return attachments.
generateGdbParams.ReturnAttachments = false;

// Create the generate geodatabase job, pass in the parameters and an output path for the local
geodatabase.
// ...
```

Create the geodatabase

Obtain a job to generate and download the geodatabase by passing the `GenerateGeodatabaseParameters` to the `GenerateGeodatabase` method on the `GeodatabaseSyncTask`. Run the job to generate and download the geodatabase to the device.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Instantiate the geodatabase sync task and define the task parameters (see previous example).

// Create the generate geodatabase job, pass in the parameters and an output path for the local
geodatabase.
GenerateGeodatabaseJob generateGdbJob = gdbSyncTask.GenerateGeodatabase(generateGdbParams,
outGeodatabasePath);

// Handle the JobChanged event to check the status of the job.
generateGdbJob.JobChanged += (sender, args) =>
{
    // If there was an error, report it and return.
    if (generateGdbJob.Error != null)
    {
        Console.WriteLine("Error creating geodatabase: " + generateGdbJob.Error.Message);
        return;
    }

    // Check the job status.
    if (generateGdbJob.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Succeeded)
    {
        // If the job succeeded, call a function to add local data to the map.
        AddLocalDataToMap(generateGdbJob);
    }
    else if (generateGdbJob.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Failed)
    {
        // If the job failed, report failure.
        Console.WriteLine("Unable to create local geodatabase.");
    }
    else
    {
        // The job is still running, report the last message.
        Console.WriteLine(generateGdbJob.Messages[generateGdbJob.Messages.Count - 1].Message);
    }
};

// Start the job and report the job ID.
generateGdbJob.Start();
Console.WriteLine("Submitted job #" + generateGdbJob.ServerJobId + " to create local
geodatabase.");
```

Your geodatabase is now on the local machine or device.

If you generate the geodatabase using these methods, it will automatically register the geodatabase with its original service. This allows the data in the geodatabase to be synchronized with the original service. If you want to load this geodatabase onto a number of devices and allow those users to synchronize changes with the original service, you must register these individual geodatabases with the original service.

Note:
For details on synchronizing offline feature edits, see Sync offline edits.

Create layers from geodatabase feature tables

Now that you have a geodatabase on your local machine or device, you'll create the relevant layer from the geodatabase feature tables.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
Geodatabases downloaded in this way can contain tables with one of two types of data: features and annotation (as of ArcGIS Runtime 100.6 and ArcGIS Enterprise 10.7.1).

1. Get the geodatabase from the generate geodatabase job.
2. Load the geodatabase.
   a. Create feature layers from the geodatabase feature tables.
   b. Create annotation layers from the geodatabase annotation tables.

```
// Get the local geodatabase from the job result.
Geodatabase localGdb = await geodatabaseJob.GetResultAsync();

// Create a new feature layer for each feature table in the local geodatabase.
foreach (FeatureTable featureTable in localGdb.GeodatabaseFeatureTables)
{
    FeatureLayer featureLayer = new FeatureLayer(featureTable);
    MyMapView.Map.OperationalLayers.Add(featureLayer);
}

// Create an annotation layer for each feature table that contains annotation.
// (Annotation is stored in a ArcGISFeatureTable rather than the more generic FeatureTable)
foreach (ArcGISFeatureTable annoFeatureTable in localGdb.GeodatabaseAnnotationTables)
{
    AnnotationLayer annoLayer = new AnnotationLayer(annoFeatureTable);
    MyMapView.Map.OperationalLayers.Add(annoLayer);
}
```

Tiled layers

Tiled layers typically provide geographical context to your maps as basemaps that display beneath your operational data. You can export and download tile caches directly to your device from any tiled data source that is export enabled. Esri provides a number of raster and vector basemaps for you to export and download:

- Download tiled data—Tiled ArcGIS map services, tiled ArcGIS image services, or tiled basemaps in ArcGIS Online
- Download vector tiled data—ArcGIS vector tile services or vector basemaps in ArcGIS Online

Download tiled data

You can take tiled data offline by exporting and downloading the tiled data as a tile package (`.tpk`) using the `ExportTileCacheTask` class. This task requires that the tiled map or image service support the creation of an offline tile cache; specifically, it must enable the `exportTiles` operation. This can be found at the bottom of the service's HTML page.

To create the tile cache, do the following:

1. Generate the default export tile cache parameters and set any of the properties.
2. Export and download the tile cache using the methods on the `ExportTileCacheTask`.

Generate default export tile cache parameters

Construct a default set of parameters (`ExportTileCacheParameters`) by passing an area of interest (polygon or envelope) along with the minimum and maximum scale thresholds to the `CreateDefaultExportTileCacheParametersAsync` method on the `ExportTileCacheTask`.

Copyright © 1995-2020 Esri. All rights reserved.

This method will return a set of parameters for the area of interest and will calculate the levels of detail (LODs) required to support the minimum and maximum scale for the service's tiling scheme. You can adjust these LOD levels or remove some before you download the cache.

```
// URL to the map service with tiles to export.
Uri tilesMapServiceUri = new Uri("https://sampleserver6.arcgisonline.com/arcgis/rest/services/
World_Street_Map/MapServer");

// Instantiate an export tiles task with the URL to the tiled service.
ExportTileCacheTask exportTileCacheTask = await
ExportTileCacheTask.CreateAsync(tilesMapServiceUri);

// Create the export tile cache parameters.
ExportTileCacheParameters exportTileParams = await
exportTileCacheTask.CreateDefaultExportTileCacheParametersAsync(
    areaOfInterest: MyMapView.VisibleArea,
    minScale: MyMapView.MapScale,
    maxScale: 0);
```

Export and download the tile cache

Obtain an export and download job by passing the `ExportTileCacheParameters` to the `ExportTileCacheJob` method on the `ExportTileCacheTask`. Run the job to download the tile cache into a tile package that is placed in the download path on the device.

```
// Destination path for the local package (.tpk file), including name.
string myDocumentsFolder = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string tileCachePath = System.IO.Path.Combine(myDocumentsFolder, "StreetMapTiles.tpk");

// Get the job.
ExportTileCacheJob exportTilesJob = exportTileCacheTask.ExportTileCache(exportTileParams,
tileCachePath);

// Handle job status change to check progress.
exportTilesJob.JobChanged += async (sender, args) =>
{
    // Report the current job status and progress.
    Console.WriteLine($"Job changed; status: {exportTilesJob.Status}, progress:
{exportTilesJob.Progress}%");

    // When the job completes successfully, display the local tile cache.
    if (exportTilesJob.Status == JobStatus.Succeeded)
    {
        // Get the local tile cache.
        TileCache cache = await exportTilesJob.GetResultAsync();

        // Use the tile cache to create an ArcGISTiledLayer.
        ArcGISTiledLayer tiledLayer = new ArcGISTiledLayer(cache);

        // Display the tiled layer as a basemap.
        MyMapView.Map = new Map(new Basemap(tiledLayer));
    }
};

// Start the export tile cache job.
exportTilesJob.Start();
```

When creating an offline tile cache from a tiled service, consider the following:

Copyright © 1995-2020 Esri. All rights reserved.

- The export tiles operation used to generate tile caches is only available with services hosted on ArcGIS Server 10.2.1 or later.
- Estimation of tile cache size is not available on ArcGIS Online hosted tiled services.
- The time required to create a tile cache varies depending on the extent requested, the number of levels requested, the network connection speed, and so on. Use the `EstimateTileCacheSize` method on the `ExportTileCacheTask` class, which returns an `EstimateTileCacheSizeJob` to get the approximate size of a tile cache for a specific set of parameters. Try generating a smaller tile cache to get an idea of how long caching will take when planning a large offline data capture event.
- There is a limit to the number of tiles you can generate in a single request. When generating a tile cache from an ArcGIS Online basemap, there is a limit of 100,000 tiles per request. Read more on this in the ArcGIS REST API documentation. Organizations that use their own ArcGIS Server to host an offline tile cache can configure the server's `maxExportTilesCount` via the administrator account to change the default limit of 100,000 tiles.

Download vector tiled data

You can take vector tiled data offline by exporting it from an ArcGIS vector tile service and downloading it as a vector tile package (`.vtpk`) using the `ExportVectorTilesTask` class. The vector tile service used for this operation must support the creation of an offline vector tile cache; specifically, it must enable the `exportTiles` operation. Vector tiles contain vector representations of data that can be restyled for different purposes, such as day and night viewing. You can download default styling resources along with the vector tiles and custom style resources from ArcGIS Portal items that host vector tile layers.

You have a number of workflows available to you depending on whether your vector tiled data has custom style resources, whether you want to download many custom styles that you can apply to a number of tile caches, or whether you just want to obtain the default tile cache.

To create a vector tile package (`.vtpk` file) and vector tile style resources, do the following:

1. Instantiate the export vector tiles task and check whether the vector tiles have custom style resources.
2. Specify the export vector tile task parameters for a specific maximum scale and area of interest.
3. Export the vector tiles. The vector tile package is populated with vector tiles and the default style resources from the ArcGIS vector tile service.

Note:
If your ArcGIS vector tile layer has custom style resources, export the vector tiles with custom style resources. If you only want to obtain the custom style resources, see export custom style resources.

Instantiate the export vector tiles task

Instantiate the `ExportVectorTilesTask` using a URL to the portal item that represents an ArcGIS vector tiled layer. Load the task, and upon completion, check whether the vector tiles have custom style resources by checking the Boolean value.

```
// Connect to a portal (ArcGIS Online) to access a portal item ("Colored pencil" vector tile
layer).
ArcGISPortal portal = await ArcGISPortal.CreateAsync();
string itemId = "4cf7e1fb9f254dcda9c8fbadb15cf0f8";
PortalItem vectorTiledLayerItem = await PortalItem.CreateAsync(portal, itemId);

// Initialize the export vector tile task with the portal item for a vector tiled layer.
// (Can use a URL if only the default styles are required).
ExportVectorTilesTask exportVectorTileTask = await
```

Copyright © 1995-2020 Esri. All rights reserved.

```
ExportVectorTilesTask.CreateAsync(vectorTiledLayerItem);

// Check if the vector tile layer has style resources.
bool hasStyleResources = exportVectorTileTask.HasStyleResources;

// Choose whether to download just the style resources or both the styles and the tiles.
if (hasStyleResources)
{
    //...
}
else
{
    //...
}
```

Note:

You can instantiate the `ExportVectorTilesTask` with either of the following:

- A URL to the portal item that represents an ArcGIS vector tiled layer. This gives you the ability to enable with the layer's custom styles. (Recommended approach)
- A URL to the ArcGIS vector tile service itself. Use this option if you only want to display the service's default style.

Specify the export vector tiles task parameters

To obtain a default set of parameters, call the `ExportVectorTilesTask.CreateDefaultExportVectorTilesParametersAsync` method and provide an area of interest (polygon or envelope) and a maximum scale. When you provide the maximum scale, you must be aware that there won't be any tiles when the map is zoomed in beyond this scale. If you set the maximum scale to 0, the export will include all levels of detail in the service.

This method returns the set of default parameters, `ExportVectorTilesParameters`, required to export the vector tiles to a vector tile package. The LODs have been calculated to support the maximum scale that you specified.

```
// Create the default export vector tile cache job parameters.
ExportVectorTilesParameters exportVectorTileParams = await
exportVectorTileTask.CreateDefaultExportVectorTilesParametersAsync(
    areaOfInterest: MyMapView.VisibleArea,
    maxScale: MyMapView.MapScale);

// Destination path for the local vector cache (.vtpk file).
string myDocumentsFolder = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string tileCachePath = System.IO.Path.Combine(myDocumentsFolder, "VectorMapTiles.vtpk");

// Create the job from the parameters and path to the local cache.
ExportVectorTilesJob exportVectorTilesJob =
exportVectorTileTask.ExportVectorTiles(exportVectorTileParams, tileCachePath);
```

Export the vector tiles

Obtain a job to generate and download a vector tile package and its default style resources by passing the `ExportVectorTilesParameters` to the `ExportVectorTiles` method on the `ExportVectorTilesTask`. You must also provide a download path to store the vector tile package and its default style resources.

Run the `exportVectorTilesJob` to export and download the vector tile package (`.vtpk`).

Copyright © 1995-2020 Esri. All rights reserved.

Note:
To obtain the vector tiles and the default style resources, you must instantiate the `ExportTileCacheTask` with a URL to the ArcGIS vector tile service.

```
// Get the URL to the vector tile layer source.
Uri vectorTileLayerUri = vectorTiledLayer.Source;

// Initialize the export vector tile task with the vector tile source (URI).
exportVectorTileTask = await ExportVectorTilesTask.CreateAsync(vectorTileLayerUri);

// Create the default export vector tile cache job parameters.
ExportVectorTilesParameters exportVectorTileParams = await
exportVectorTileTask.CreateDefaultExportVectorTilesParametersAsync(
    areaOfInterest: MyMapView.VisibleArea,
    maxScale: MyMapView.MapScale);

// Destination path for the local vector cache (.vtpk file).
string myDocumentsFolder = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string tileCachePath = System.IO.Path.Combine(myDocumentsFolder, "VectorMapTiles.vtpk");

// Create the job from the parameters and path to the local cache.
ExportVectorTilesJob exportVectorTilesJob =
exportVectorTileTask.ExportVectorTiles(exportVectorTileParams, tileCachePath);

// Handle job status change to check the status.
exportVectorTilesJob.JobChanged += async (sender, args) =>
{
    // Show job status and progress.
    Console.WriteLine($"Job status: {exportVectorTilesJob.Status}, progress:
{exportVectorTilesJob.Progress}%");

    // When the job succeeds, display the local vector tiles.
    if (exportVectorTilesJob.Status == JobStatus.Succeeded)
    {
        // Get the result from the job.
        ExportVectorTilesResult result = await exportVectorTilesJob.GetResultAsync();

        // Create a vector tile cache from the result.
        VectorTileCache vectorCache = result.VectorTileCache;

        // Create new vector tiled layer using the tile cache.
        ArcGISVectorTiledLayer localVectorTileLayer = new ArcGISVectorTiledLayer(vectorCache);

        // Display the layer as a basemap.
        MyMapView.Map = new Map(new Basemap(localVectorTileLayer));
    }
};

// Start the job.
exportVectorTilesJob.Start();
```

Export the vector tiles with custom style resources

Obtain a job to generate and download a vector tile package containing tiles and associated style resources by passing the `ExportVectorTilesParameters` to the `ExportVectorTiles` method on the `ExportVectorTilesTask`. The portal item's associated style resources will be downloaded and saved separately. You must also provide a download path to store the vector tile package and a separate download path for the style resources.

Copyright © 1995-2020 Esri. All rights reserved.

Run the job to export and download the vector tile package (`.vtpk`) and the style resources to the device.

Note:
To obtain the vector tile package and its custom style resources, you must instantiate the `ExportVectorTilesTask` with a portal item that represents an ArcGIS vector tiled layer.

```
// Get the portal item for the layer.
string vectorTiledLayerItemId = vectorTiledLayer.Item.ItemId;
PortalItem vectorTiledLayerItem = await PortalItem.CreateAsync(portal, vectorTiledLayerItemId);

// Initialize the export vector tile task with the portal item for a vector tiled layer.
ExportVectorTilesTask exportVectorTileTask = await
ExportVectorTilesTask.CreateAsync(vectorTiledLayerItem);

// Create the default export vector tile cache job parameters.
ExportVectorTilesParameters exportVectorTileParams = await
exportVectorTileTask.CreateDefaultExportVectorTilesParametersAsync(
    areaOfInterest: MyMapView.VisibleArea,
    maxScale: MyMapView.MapScale);

// Destination path for the local vector cache (.vtpk file) and style resources.
string myDocumentsFolder = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string tileCachePath = System.IO.Path.Combine(myDocumentsFolder, "VectorMapTiles.vtpk");
string resourcePath = System.IO.Path.Combine(myDocumentsFolder, "styleResources");

// Create the job from the parameters and paths to the local cache and resources.
ExportVectorTilesJob exportVectorTilesJob =
exportVectorTileTask.ExportVectorTiles(exportVectorTileParams, tileCachePath, resourcePath);

// Handle job status change to check the status.
exportVectorTilesJob.JobChanged += async (sender, args) =>
{
    // Show job status and progress.
    Console.WriteLine($"Job status: {exportVectorTilesJob.Status}, progress:
{exportVectorTilesJob.Progress}%");

    // When the job succeeds, display the local vector tiles.
    if (exportVectorTilesJob.Status == JobStatus.Succeeded)
    {
        // Get the result from the job.
        ExportVectorTilesResult result = await exportVectorTilesJob.GetResultAsync();

        // Create a vector tile cache and item resource cache from the result.
        VectorTileCache vectorCache = result.VectorTileCache;
        ItemResourceCache resourceCache = result.ItemResourceCache;

        // Create new vector tiled layer using the tile and resource caches.
        ArcGISVectorTiledLayer vectorTileLayer = new ArcGISVectorTiledLayer(vectorCache,
resourceCache);

        // Display the layer as a basemap.
        MyMapView.Map = new Map(new Basemap(vectorTileLayer));
    }
};

// Start the job.
exportVectorTilesJob.Start();
```

Copyright © 1995-2020 Esri. All rights reserved.

Export custom style resources

Obtain a job to download any custom style resources associated with the tasks vector tiles by passing a download path to the `ExportStyleResourceCache` method on the `ExportVectorTilesTask`.

Run the job to export the style resources. Obtain the `ItemResourceCache` from the `ExportVectorTilesResult`.

Note:
To obtain the custom style resources, you must instantiate the `ExportTileCacheTask` with a portal item that represents an ArcGIS vector tiled layer.

```
// Get the portal item for the layer.
string vectorTiledLayerItemId = vectorTiledLayer.Item.ItemId;
PortalItem vectorTiledLayerItem = await PortalItem.CreateAsync(portal, vectorTiledLayerItemId);

// Initialize the export vector tile task with the portal item for a vector tiled layer.
ExportVectorTilesTask exportVectorTileTask = await
ExportVectorTilesTask.CreateAsync(vectorTiledLayerItem);

// Destination path for the local style resources.
string myDocumentsFolder = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
string resourcePath = System.IO.Path.Combine(myDocumentsFolder, "styleResources");

// Create the job from the path to the local resources.
ExportVectorTilesJob exportStylesJob =
exportVectorTileTask.ExportStyleResourceCache(resourcePath);

// Handle job status change to check the status.
exportStylesJob.JobChanged += async (sender, args) =>
{
    // Show job status and progress.
    Console.WriteLine($"Job status: {exportStylesJob.Status}, progress:
{exportStylesJob.Progress}%");

    // When the job succeeds, access the local styles.
    if (exportStylesJob.Status == JobStatus.Succeeded)
    {
        // Get the result from the job.
        ExportVectorTilesResult result = await exportStylesJob.GetResultAsync();

        // Get stuff from the resource cache.
        ItemResourceCache resourceCache = result.ItemResourceCache;
        // ...
    }
};

// Start the job.
exportStylesJob.Start();
```

Copyright © 1995-2020 Esri. All rights reserved.

Take a scene offline - ArcGIS Pro

You can use ArcGIS Pro to consolidate your scenes into a single mobile scene package (`.mspk`) that can be taken offline. Each package can contain scenes, elevation sources and any data necessary for your users to continue working when connectivity is poor or non-existent. Mobile scene packages can include basemaps, operational layers, 3D scene layers, tiled layers (`.tpkx`,`.vtpk`), tables, relationship classes, locators, and elevation sources. Each package adheres to a common scene definition, allowing you to transport your scenes and data across the ArcGIS platform.

With ArcGIS Pro 2.6 (or later) you can include online layers and tables in your scenes, such as traffic information, weather conditions, or a large company data-set. If your user's device has network connectivity they can take advantage of these online layer. If the connection is unavailable, users can continue to work with the scenes and data that is stored locally in the mobile scene package.

Create a mobile scene package with ArcGIS Pro

1. Locate the ArcGIS Pro tool called Create Mobile Scene Package in the Data Management Toolbox. You can use this to create a mobile scene package containing any number of scenes.
2. If needed, set an expiration date on the mobile scene package. This is available with ArcGIS Pro 2.4 (or later) with the ArcGIS Publisher extension license.
3. If you want include online layers and tables in the scenes, you need to check on the "Reference online content". This is available with ArcGIS Pro 2.6 (or later).
4. Save the package to ArcGIS Online, ArcGIS Enterprise, or to a folder on your device.
5. You can download the mobile scene package from ArcGIS Online or ArcGIS Enterprise directly onto your device using the ArcGIS Runtime API. Alternatively, you can copy the package directly to your device.

Open the mobile scene package

To create the mobile scene package object, pass the mobile scene package file path `MobileScenePackage` constructor. Load the package and access its collection of scenes. Passing the scene to the scene view will initiate loading the scene, its layers, and data.

You can access the metadata from the mobile scene package, without loading the scenes from the `MobileScenePackage.Item` property. After loading a scene, its metadata is accessible using the `Scene.Item` property.

Note:

If the mobile scene package was created with ArcGIS Pro 2.6 (or later) it may contain online layers or tables. In this case your app may need to handle authentication before the layers and tables are loaded.

If the mobile scene package was created with ArcGIS Pro 2.4 (or later) it can have an expiration date. If you do not handle the potential expiration the mobile scene package may fail to load.

Handle mobile scene package expiration

Mobile scene packages, created with ArcGIS Pro 2.4 (or later), can have an expiration date set. This gives the author more control over the lifetime of the package's data. For example, it will allow you to do the following:

- Provide data to third-party contractors that will expire at the end of a contract.
- Indicate to your users that the data is now obsolete.

Copyright © 1995-2020 Esri. All rights reserved.

Use the ArcGIS Pro tool, create a mobile scene package to set an expiration date/time, an expiration message, and data access options. Setting expiration requires an ArcGIS Pro Publisher extension license.

As an ArcGIS Runtime developer, there are three possible outcomes when you try to load a mobile scene package:

1. You have full access to the mobile scene package and their scenes. This situation occurs if the package has been created without any expiration properties or if it has not expired.
2. You have full access to the mobile scene package and their scenes, even though the scenes and their data have expired. You will be warned that the package is out of date.
3. You cannot access the mobile scene package and their scenes because it has expired. In this case, the package will fail to load and you will not be able to access its scenes and data.

These three outcomes are managed by the mobile scene package load process `MobileScenePackage.Load` . No extra workflow or code paths are required to accommodate this.

If you want to provide your users with extra information, you can retrieve the expiration details from the `expiration` property on the `MobileScenePackage`. For example, you could display how many days remain before the package expires by doing the following:

1. Load the mobile scene package.
2. Read the expiration property on the mobile scene package.
3. Confirm that it has not expired.
4. Get the expiration date.
5. Calculate the number of days between today and the expiration date. Report the number of days remaining.

Copyright © 1995-2020 Esri. All rights reserved.

Display information

Copyright © 1995-2020 Esri. All rights reserved.

Add graphics overlays to your app

A graphics overlay is a container for temporary graphics that you want to display on your map view or scene view, such as the graphics in the following illustration. The graphics you draw in your graphics overlays are created at run time and are not persisted when your application closes.

Typical use cases for adding graphic overlays to your map view or scene view include the following:

- Showing the location of a moving object. The locations could be from your own GPS receiver
- Showing the results of a geocoding or routing task
- Drawing temporary map markups, sometimes referred to as red lining

A map view or scene view can contain many graphics overlays that are drawn on top of the layers in your map or scene. Each graphics overlay may contain a number of graphics. A graphic is defined by the following components:

- Geometry, which is either a point, line, or polygon
- Symbology, which defines how your geometry is drawn. For example, if the geometry is a line, then your symbology would define the line color as red
- Attributes, which are an optional set of item fields in a database row

The following code shows how to create a new graphics overlay and add it to your map view.

```
// create a new graphics overlay
var graphicsOverlay = new GraphicsOverlay();

// add the overlay to the map view's graphics overlay collection
myMapView.GraphicsOverlays.Add(graphicsOverlay);
```

Now you're ready to add graphics and text to your graphics overlays.

Note:

Graphics overlays are visible only on a map view which contains a map, or a scene view which contains a scene. Graphics overlays do not have a time extent and do not participate in time-based filtering when a time extent is set on a map view or scene view.

Copyright © 1995-2020 Esri. All rights reserved.

Related topics

Add features and graphics to a scene view
Add graphics and text to graphics overlays
Features and graphics
Symbols, renderers, and styles

Copyright © 1995-2020 Esri. All rights reserved.

Add graphics and text to graphics overlays

A graphic is a visible item on your map that can be seen as a point, line, polygon, or text. You can add graphics to your map view. These are contained in graphics overlays and are for showing temporary data that's not persisted once the application closes. For example, you may want to display a route between two locations or animate data items that change quickly.

This topic discusses adding graphics to your map in a way that generates a symbol instance for each graphic. If you're drawing a large number of graphics with the same symbology, then you should consider applying a renderer to the graphics overlay instead, which allows you to define the appearance of all the graphics in the overlay. Using renderers to add graphics is discussed in Symbols and renderers.

The image below shows an example of graphics drawn over a gray basemap. The graphics are showing:

- Red markers showing location of buoys
- Text symbols adding names to the islands of Craigleith and Bass Rock
- A green cross hatched polygon showing a Gannet nesting ground on Bass Rock
- A purple line showing the path of a boat trip.

For more information on graphics and when to use them, see Features and graphics. For more information on graphics overlays, see Add graphics overlays to your app.

Add point graphics

The following steps show how to create the red buoy markers on the image above.

1. The location of point graphics is represented by the `MapPoint` geometry class. The following code shows how the buoy locations are defined:

```
// define the buoy locations
var wgs84 = SpatialReferences.Wgs84;
var buoy1Loc = new MapPoint(-2.712642647560347, 56.062812566811544, wgs84);
var buoy2Loc = new MapPoint(-2.6908416959572303, 56.06444173689877, wgs84);
var buoy3Loc = new MapPoint(-2.6697273884990937, 56.064250073402874, wgs84);
var buoy4Loc = new MapPoint(-2.6395150461199726, 56.06127916736989, wgs84);
```

   The points are defined in the WGS84 spatial reference, which is generally the spatial reference you would obtain from a GPS receiver, for example.

Copyright © 1995-2020 Esri. All rights reserved.

Tip:

If the spatial reference of the geometry of a graphic is null, then it is assumed the geometry is in the same spatial reference as the map view; if your graphics do not display as expected and have a null spatial reference, check that the coordinates of your geometry use the same spatial reference as the map view. If the spatial reference is set, and is different to that of the map view, geometries are automatically reprojected for display. For maximum efficiency, especially when dealing with large numbers of graphics or moving graphics, geometries of graphics should have the same spatial reference as the map view.

2. Define what the points look like. Points can be drawn as simple marker symbols (circles, crosses, diamonds, etc.) or picture marker symbols represented by small bitmap images.
   The following code shows how to define a red circle marker symbol.

```
// create a marker symbol
var buoyMarker = new SimpleMarkerSymbol(SimpleMarkerSymbolStyle.Circle,
System.Drawing.Color.Red, 10);
```

3. Create graphics, which are defined by the point geometry and the symbol.

```
// create graphics
var buoyGraphic1 = new Graphic(buoy1Loc, buoyMarker);
var buoyGraphic2 = new Graphic(buoy2Loc, buoyMarker);
var buoyGraphic3 = new Graphic(buoy3Loc, buoyMarker);
var buoyGraphic4 = new Graphic(buoy4Loc, buoyMarker);
```

4. Add the graphics to the graphics overlay, which will display them on the map view.

```
// Add the graphics to the graphics overlay
MyGraphicsOverlay.Graphics.Add(buoyGraphic1);
MyGraphicsOverlay.Graphics.Add(buoyGraphic2);
MyGraphicsOverlay.Graphics.Add(buoyGraphic3);
MyGraphicsOverlay.Graphics.Add(buoyGraphic4);
```

Add line graphics

The following steps show how to create the purple, dashed line representing the boat trip in the image near the top of this topic.

1. The geometry for the boat trip is represented by the `Polyline` class. This is built with a collection of points used to define the vertices of the line.

```
//  use a polyline builder to create the new polyline from a collection of points
var boatPositions = new PolylineBuilder(SpatialReferences.Wgs84);

//  add positions to the builder
boatPositions.AddPoint(new MapPoint(-2.7184791227926772, 56.06147084563517));
boatPositions.AddPoint(new MapPoint(-2.7196807500463924, 56.06147084563517));
boatPositions.AddPoint(new MapPoint(-2.722084004553823, 56.062141712059706));
boatPositions.AddPoint(new MapPoint(-2.726375530459948, 56.06386674355254));
// ... more points here ...
boatPositions.AddPoint(new MapPoint(-2.7167625124302273, 56.06453756828941));
boatPositions.AddPoint(new MapPoint(-2.718307461756433, 56.06348340989081));
boatPositions.AddPoint(new MapPoint(-2.719165766937657, 56.062812566811544));
boatPositions.AddPoint(new MapPoint(-2.7198524110826376, 56.06204587471371));
boatPositions.AddPoint(new MapPoint(-2.719165766937657, 56.06166252294756));
boatPositions.AddPoint(new MapPoint(-2.718307461756433, 56.06147084563517));
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// get the polyline from the builder
var boatRoute = boatPositions.ToGeometry();
```

For more information on drawing lines using the geometry API, see Geometries.

2. Define the symbol for the boat trip using the `SimpleLineSymbol` class.

```
// create a simple line symbol to display the polyline
var lineSymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Dash, Color.FromArgb(255, 128, 0,
128), 1.0);
```

3. Create the graphic from the line geometry and line symbol.

```
// create the graphic
var boatTripGraphic = new Graphic(boatRoute, lineSymbol);
```

4. Add the graphic to the graphics overlay.

```
// add the polyline graphic
MyGraphicsOverlay.Graphics.Add(boatTripGraphic);
```

Add polygon graphics

The following steps show how to create the polygon representing the nesting ground area in the image near the top of this topic.

1. The geometry for the nesting ground is defined using the `Polygon` class. The vertices of the polygon are created from a collection of points. For more information on building polygons, see Geometries.

```
// use a polygon builder to create a new polygon from a collection of points
var polygonBuilder = new PolygonBuilder(SpatialReferences.Wgs84);

// add points to the polygon builder
polygonBuilder.AddPoint(new MapPoint(-2.643077012566659, 56.077125346044475));
polygonBuilder.AddPoint(new MapPoint(-2.6428195210159444, 56.07717324600376));
polygonBuilder.AddPoint(new MapPoint(-2.6425405718360033, 56.07774804087097));
polygonBuilder.AddPoint(new MapPoint(-2.6427122328698127, 56.077927662508635));
polygonBuilder.AddPoint(new MapPoint(-2.642454741319098, 56.07829887790651));
// ... more points here ...
polygonBuilder.AddPoint(new MapPoint(-2.6408883343855822, 56.075808073830935));
polygonBuilder.AddPoint(new MapPoint(-2.6417680971838577, 56.076239186057734));
polygonBuilder.AddPoint(new MapPoint(-2.642197249768383, 56.076251161328514));
polygonBuilder.AddPoint(new MapPoint(-2.6428409786451708, 56.07661041772168));
polygonBuilder.AddPoint(new MapPoint(-2.643077012566659, 56.077125346044475));

// get the polygon from the polygon builder
var nestingGround = polygonBuilder.ToGeometry();
```

2. Define a symbol for the polygon. The symbol in the screen shot above is made up of a line symbol for the outline and a fill symbol for the polygon fill.

```
// define the polygon outline
var outlineSymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Dash, Color.Blue, 1.0);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// define the fill symbol
var fillSymbol = new SimpleFillSymbol(SimpleFillSymbolStyle.DiagonalCross,
Color.FromArgb(255, 0, 80, 0), outlineSymbol);
```

3. Create the graphic from the polygon geometry and fill symbol.

```
// create the graphic
var nestingGraphic = new Graphic(nestingGround, fillSymbol);
```

4. Add the polygon graphic to the graphics overlay.

```
// add the polygon graphic
MyGraphicsOverlay.Graphics.Add(nestingGraphic);
```

Add text

The following steps show how to add the "Craigleith" and "Bass Rock" text to the graphics overlay.

1. The geometry for text symbols is a point. The point represents the anchor point for the text. The following code defines the anchor points for the two text symbols in the image at the top of this topic.

```
var labelPointBass = new MapPoint(-2.640631, 56.078083, SpatialReferences.Wgs84);
var labelPointCraigleith = new MapPoint(-2.720324, 56.073569, SpatialReferences.Wgs84);
```

2. Create the symbols for the text items.

```
// create text symbols
var textSymBass = new TextSymbol("Bass Rock",
                                 Color.FromArgb(255, 0, 0, 230),
                                 10,
                                 Esri.ArcGISRuntime.Symbology.HorizontalAlignment.Left,
                                 Esri.ArcGISRuntime.Symbology.VerticalAlignment.Bottom);
var textSymCraigleith = new TextSymbol("Craigleith",
                                 Color.FromArgb(255, 0, 0, 230),
                                 10,
                                 Esri.ArcGISRuntime.Symbology.HorizontalAlignment.Right,
                                 Esri.ArcGISRuntime.Symbology.VerticalAlignment.Top);
```

3. Create the graphics from the point and text symbol.

```
// create graphics using the points and text symbols
var graphicBass = new Graphic(labelPointBass, textSymBass);
var graphicCraigleith = new Graphic(labelPointCraigleith, textSymCraigleith);
```

4. Add the text graphics to the graphics overlays.

```
// add text graphics to the overlay
MyGraphicsOverlay.Graphics.Add(graphicBass);
MyGraphicsOverlay.Graphics.Add(graphicCraigleith);
```

Copyright © 1995-2020 Esri. All rights reserved.

Identify graphics

In your app, you can add code to allow users to click on a graphic and get more information about it. For example, you may tap or click on a seabird in your graphics overlay to display the name of the bird's species in a dialog box.

The `IdentifyGraphicsOverlayAsync` method on the map view allows you identify graphics at a point on the map for a given graphics overlay. If you have more than one graphics overlay and want to perform an identify operation on all graphics overlays, use the `IdentifyGraphicsOverlaysAsync` method. The example below shows how you can perform an identify on a single graphics overlay in response to a tap on the map view.

```
// identify graphics in the MyGraphicsOverlay overlay using the point tapped
var resultGraphics = await MyMapView.IdentifyGraphicsOverlayAsync(MyGraphicsOverlay, e.Position,
10, 2);

// show attribute information for the first graphic identified (if any)
if (resultGraphics != null && resultGraphics.Count > 0)
{
    var idGraphic = resultGraphics.FirstOrDefault();

    NameTextBox.Text = idGraphic.Attributes["Name"].ToString();
}
```

You can iterate through each result returned from an identify. For example, you can display a list of the attribute values from each result.

```
// show the name attribute for all the graphics identified (if any)
if (resultGraphics != null && resultGraphics.Graphics.Count > 0)
{
    foreach(var g in resultGraphics.Graphics)
    {
        NameListBox.Items.Add(g.Attributes["Name"]);
    }
}
```

Select graphics

Selecting a graphic is a way of bringing it to the attention of a user by drawing a halo around it. Change the selection state of a graphic by using the `IsSelected` property on the `Graphic` class.

Copyright © 1995-2020 Esri. All rights reserved.

Update graphics

It's possible to update the geometry or attributes of a graphic by using the graphics collection of the graphics overlay.

```
// get the current location (point) of the graphic
var currentPosition = boatGraphic.Geometry as MapPoint;

// define new x and y coordinates by applying an offset
var newX = currentPosition.X + deltaX;
var newY = currentPosition.Y + deltaY;

// update the point with the new coordinates (graphic will update to show new location)
var updatedPosition = new MapPoint(newX, newY);
boatGraphic.Geometry = updatedPosition;
```

Remove graphics

To remove graphics you work with the graphics collection and call the `Remove` method.

```
MyGraphicsOverlay.Graphics.Remove(graphicBass);
// ... or use RemoveAt to remove a graphic using its index
MyGraphicsOverlay.Graphics.RemoveAt(0);
```

Related topics

Add features and graphics to a scene view
Add graphics overlays to your app
Features and graphics
Symbols, renderers, and styles

Copyright © 1995-2020 Esri. All rights reserved.

Add image overlays

Image overlays can be used to quickly render frequently changing images in a scene view. For example, rendering realtime sensor data, such as weather, where each static image represents a single frame from the radar. Similar to graphics overlays, which contain a collection of graphics based on geometry (points, lines, or polygons), image overlays manage a collection of image frames.

Image overlays and graphics overlays both render above all layers in the scene. Image overlays render below any graphics overlays you've added to the scene view. This ensures that graphics you want to display won't be obscured by your image overlays. For example, you might want to show graphics representing aircraft on top of image overlays showing the sensor data they are collecting.

Since they are designed for quick display, image overlays do not support the rich processing and rendering capabilities of a raster layer, which still provides the best option for workflows that require static image rendering, analysis, and persistence. See Add raster data for more information about working with raster layers.

Note:
Image overlays can only be added to a scene view. Map view does not currently support image overlays.

Add an image overlay to a scene view

A scene view manages a collection of image overlays. Each image overlay contains a single image frame that defines an image to display. Georeferenced images (those with a world file) are added at the correct geographic location. Otherwise, you must define a geographic extent for the image frame. If a spatial reference is not defined for the extent, it is assumed to be the same as the scene. If the spatial reference of the extent is different from that of the scene, the image will fail to render.

1. If your image doesn't have georeference information, define an extent for the image using an envelope or quadrilateral polygon.
   The following example defines a geographic extent using an envelope defined with a center point, width, and height. You can also specify an extent using a polygon as long as it has exactly four points (to define each corner of the image).

```
// Create an Envelope for displaying the image frame in the correct location
MapPoint centerPoint = new MapPoint(-120.0724, 35.1310, SpatialReferences.Wgs84);
double width = 15.0959;
double height = 14.3770;
Envelope pacificSouthwestEnvelope = new Envelope(centerPoint, width, height);
```

2. Create an image frame and pass an image into the constructor. You can specify an image with a `URI` to a local or online source, supported formats are PNG, JPEG, TIFF, and GeoTIFF. If you need to define the extent explicitly, also pass the envelope or polygon to the constructor.

   Note:
   You can also specify an image by passing a `RuntimeImage` to the constructor. These formats are supported for `RuntimeImage`: BMP, GIF, JPG, PNG, and ICO. Animated GIF is not supported, but can be displayed as a static image.

```
// Create a RuntimeImage to display using a local png file
RuntimeImage image = new RuntimeImage(new System.Uri("file:///c:/radar/
pacsouthwest_radaronly.png"));
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create an ImageFrame with a local image file and the extent envelope
ImageFrame imageFrame = new ImageFrame(image, pacificSouthwestEnvelope);
```

3. Add the image frame to an image overlay. The image overlay has properties to control visibility and opacity of the image. Set the opacity less than 1.0 to make the image semi-transparent so data underneath can be seen.

```
// Add the ImageFrame to an ImageOverlay and set it to be 50% transparent
ImageOverlay imageOverlay = new ImageOverlay(imageFrame);
imageOverlay.Opacity = 0.5;
```

4. Add the image overlay to the scene view's image overlay collection. To zoom to the extent of the image, you can set the scene view's viewpoint using the image frame extent.

```
// Add the ImageOverlay to the scene view's ImageOverlay collection
MySceneView.ImageOverlays.Add(imageOverlay);

// Set the viewpoint with the ImageFrame's extent
await MySceneView.SetViewpointAsync(new Viewpoint(imageFrame.Extent));
```

Note:
Like the graphics overlays collection, a scene view image overlays collection can contain multiple overlays. Both types of overlays draw over data in the scene (base maps, operational layers, and so on). Graphics overlays display on top of image overlays in the view.

Note:
Radar data for the United States, like that shown in the preceding image, are available for download from the National Weather Service web site.

Animate an image overlay

You can animate the display of image overlays by changing the frame they contain at a specified interval. You might use a timer, for example, to read the next image in a sequence, use it to create a new image frame, and replace the current frame in the image overlay.

Note:
Animated GIF format is not supported. If added to an image overlay, it will display as a static image.

Copyright © 1995-2020 Esri. All rights reserved.

1. Create a collection of images in the correct sequence. These may be added explicitly, read from a local folder, downloaded from a service, or perhaps read from a shared location that is periodically updated with new data.
   The example below creates a sequence of images and adds them to a list.

```
// Create an array of images to display in sequence
_radarImages = new List<RuntimeImage>();
RuntimeImage image = new RuntimeImage(new System.Uri("file:///c:/radar/
pacsouthwest_radaronly.png"));
RuntimeImage image1 = new RuntimeImage(new System.Uri("file:///c:/radar/
pacsouthwest_radaronly1.png"));
RuntimeImage image2 = new RuntimeImage(new System.Uri("file:///c:/radar/
pacsouthwest_radaronly2.png"));
RuntimeImage image3 = new RuntimeImage(new System.Uri("file:///c:/radar/
pacsouthwest_radaronly3.png"));
RuntimeImage image4 = new RuntimeImage(new System.Uri("file:///c:/radar/
pacsouthwest_radaronly4.png"));
_radarImages.Add(image);
_radarImages.Add(image1);
_radarImages.Add(image2);
_radarImages.Add(image3);
_radarImages.Add(image4);
```

2. Create a new timer and set it to call a function to change the image at the desired interval.
   The following example creates a timer that calls a function to change the image in the overlay every second.

```
// Create a timer
_timer = new DispatcherTimer(DispatcherPriority.Render)
{
    // Fire the 'tick' event every second
    Interval = TimeSpan.FromMilliseconds(1000),
    IsEnabled = true
};
// When the tick event fires, call a function to change the image
_timer.Tick += ChangeImageFrame;
```

3. For each timer interval, create a new frame with the next image in the sequence and replace the current image frame in the overlay.
   The following example changes the image in the overlay while keeping it at the same spatial extent. If required by your workflow, you could also animate the overlay location by changing the image frame's extent.

```
private void ChangeImageFrame(object sender, EventArgs e)
{
    // If all images have been shown, start back at the first
    if(_index > _radarImages.Count - 1)
    {
       _index = 0;
    }

    // Get the next image in the sequence
    RuntimeImage nextImage = _radarImages[_index];

    // Get the extent from the current frame
    Envelope ext = _imageOverlay.ImageFrame.Extent;

    // Create a new image frame with the new image and the same extent
    ImageFrame frame = new ImageFrame(nextImage, ext);

    // Replace the image frame in the image overlay
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    _imageOverlay.ImageFrame = frame;

    // Increment the image index
    _index++;
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Symbolize data

In ArcGIS Runtime, for many types of geographic data, you can control how your data are displayed. Feature layers, raster layers, graphic overlays, and even vector basemaps provide the ability to alter display characteristics like size, color, rotation and so on.

The rendering of geographic data is generally controlled by one or more of the following types of classes:

- **Symbol**—Defines a set of display properties, such as color and size, for features and graphics (collectively referred to as geoelements).
- **Renderer**—A collection of one or more symbols and logic to determine which symbol to apply to each geoelement, usually based on one or several attribute values.
- **Raster renderer**—Used exclusively for raster data, these renderers determine which color to apply for each raster cell value.
- **Style**—A predefined definition, available for some layer formats, of how a layer should be displayed. If more than one style is available, it can be changed dynamically.

The Symbols, renderers, and styles topic provides an overview of these classes. This topic describes how to customize the display of various types of data you might include in your app, and shows examples.

The following table describes options for symbolizing different types of data.

Data type	Symbology options
Graphic	• Assign an individual simple symbol or multilayer symbol to the graphic • Create a 3D symbol and assign it to the graphic for rendering in a scene view • Apply a renderer to the graphics overlay • Assign 3D properties (such as an extrusion expression) to the renderer for display in a scene view
Feature	• Apply a renderer to the feature layer or feature collection • Assign 3D properties (such as an extrusion expression) to the renderer for display in a scene view
Raster	• Apply a raster renderer to the raster layer • Apply a raster rendering rule to the raster
Vector basemap	• Apply a predefined ArcGIS vector tiled layer style • Create a custom style using the ArcGIS Vector Tile Style Editor
WMS layer	• Apply a predefined WMS Style

Work with symbols

ArcGIS Runtime SDK provides a variety of symbols for displaying geoelements in a map or scene. These symbols fall into one of two broad categories: simple or multilayer. You can create these symbols new, read them from a file, and make edits to their properties. See Symbol types described for an overview and a discussion of the available types of symbols.

Copyright © 1995-2020 Esri. All rights reserved.

Create simple symbols

Simple symbols follow the web map specification and you can create and work with them through the simple symbol classes in the API (`SimpleMarkerSymbol`, `SimpleLineSymbol`, and `SimpleFillSymbol`). These are also the symbols you get from web maps or from feature services when advanced symbology is turned off. Each of the simple symbol types provides an enumeration of pre-defined styles that can be applied to the symbol.

The following example creates a simple marker symbol and a text symbol to display on the map.

```
// create a red circle marker symbol
var markerSym = new SimpleMarkerSymbol
{
    Style = SimpleMarkerSymbolStyle.Circle,
    Color = Color.Red,
    Size = 12
};

// create a text symbol
var textSym = new TextSymbol
{
    Angle = -55,
    Color = Color.Red,
    Size = 18,
    Text = "You are here",
    HorizontalAlignment = Esri.ArcGISRuntime.Symbology.HorizontalAlignment.Left,
    VerticalAlignment = Esri.ArcGISRuntime.Symbology.VerticalAlignment.Bottom
};

// create a graphic that uses the marker symbol
var pointGraphic = new Graphic(point, markerSym);

// create a graphic that uses the text symbol
var textGraphic = new Graphic(point, textSym);

// add the graphics to a graphics overlay in the map view
var graphics = MyMapView.GraphicsOverlays[0].Graphics;
graphics.Add(pointGraphic);
graphics.Add(textGraphic);
```

There are various ways to create picture marker symbols, as the following examples illustrate.

You can create a picture marker symbol from a URL:

```
// create a new PictureMarkerSymbol with an image file stored online (.png, .jpg, .bmp)
var picUrl = @"http://sampleserver6.arcgisonline.com/arcgis/rest/services/Recreation/
FeatureServer/0/images/e82f744ebb069bb35b234b3fea46deae";
PictureMarkerSymbol picMarkerSym = new PictureMarkerSymbol(new Uri(picUrl));

// apply the symbol to a new graphic and add it to a graphics overlay in the map view
var graphic = new Graphic(location, picMarkerSym);
MyMapView.GraphicsOverlays[0].Graphics.Add(graphic);
```

Copyright © 1995-2020 Esri. All rights reserved.

You can create a picture marker symbol from a resource in the project:

```
// create a new PictureMarkerSymbol with an image resource in the project
var stream =
Helpers.ResourceLoader.GetEmbeddedResourceStream(Assembly.GetAssembly(typeof(Helpers.ResourceLoader)),
"campsite.png");
var pictureMarkerSym = await PictureMarkerSymbol.CreateAsync(stream);

// apply the symbol to a new graphic and add it to a graphics overlay in the map view
var graphic = new Graphic(location, pictureMarkerSym);
MyMapView.GraphicsOverlays[0].Graphics.Add(graphic);
```

You can create a picture marker symbol from a raw Json string:

```
// create a new PictureMarkerSymbol from a Json string
var picJson = "{\"type\":\"esriPMS\"," +

"\"imageData\":\"iVBORw0KGgoAAAANSUhEUgAAACAAAAAgCAYAAABzenr0AAAABHNCSVQICAgIfAhkiAAAAldJREFUWIXtl"
+
    "81O20AQx/dL6ZYcQpCsRBhQVLxxiWxz4IDKoRwIL8CJcw48RpA4BUGUiAOQhEbm0IAQkht4qxzKG/
DvoWLJAhHFBCykjrQHyzM" +
    "7v50Ze2cI+ciyubmJjY0NXF9d492dl9fK4JyDUYaVbyuxAURcw8FgQG5vbwkhhPy+uYm7DWFxDUEw/
JAAAO69UpoAwLhkTBGIH" +
    "4KPGwFTEojA/yI0IpBECgxJPAVJALxpCg4ODl74d38eYG9v7/
k9Ly4u4DgOOOewLAutVmukkeu6oJSCUgrfD0bq1et1TE5OgnM" +
    "O13VxeXk5GmR3dxdSSr2xZVkIw/
BJg2GAwF98UqfZbCKTyWi9dDqNRqMxGiCKIvi+DyGENioUCmi324+MDIDgMUCtVkMulwNjD" +
    "JRSCCGwtLT0fBqOjo7huq6G4JxDKYXDw0PDuFgsaoDFBwA7OzuwbdtwHgQBTk9P/
6222sdtzH+ZB2MMjDGdv2EIA2DxHmB7exszM" +
    "zPaOeccnueh1zt7WWF3u11MT0+DMQZCCIQQUErpwlRKPQKoVqvI5/PahjEGpRSiKIrXM/
V+9jDxeUKfhjGGubk5bG1tYXZ21qiT" +

"SqWCbDZr6OZz+dd3y+dn57BtG5xz7VAIoR3dOXv4vlQqja9VD8MQjuPowiSEaGcPVyr1CcvLy+OfE36cdLHwdcH4RIcXIQRSSq"
+
    "x+X327IeWkcwLP84xw3y0pJcpr5befkDqdDkqlkgEhpcT6+vr7jWdhN4Qz//fOSKVSrzp57Hu0/6uP/lWfTE1Nkf39/
Vd0BAnLH" +
    "3KRNn8UVI54AAAAAElFTkSuQmCC\"," +
    "\"contentType\":\"image/png\"," +
    "\"width\":0,\"height\":0,\"angle\":0," +
    "\"xoffset\":0,\"yoffset\":0," +
    "\"xscale\":1,\"yscale\":1}";
// pass the URI to the image to the symbol constructor
var pictureMarkerSym = Symbol.FromJson(picJson);

// apply the symbol to a new graphic and add it to a graphics overlay in the map view
var graphic = new Graphic(location, pictureMarkerSym);
MyMapView.GraphicsOverlays[0].Graphics.Add(graphic);
```

Copyright © 1995-2020 Esri. All rights reserved.

Create multilayer (advanced) symbols

Multilayer symbols are based on a subset of the ArcGIS Pro symbology model. These symbols are composed of one or more symbol layers and can contain different symbol types within each layer with definable behaviors to get advanced cartographic effects. You can modify properties of individual symbol layers, or use properties of the symbol itself to apply a value (color or size, for example) to all the symbol layers it contains.

There are multilayer symbol classes for displaying points (`MultilayerPointSymbol`), polylines (`MultilayerPolylineSymbol`), and polygons (`MultilayerPolygonSymbol`). A multilayer symbol can contain one or several symbol layers of the same or different type. A symbol used to render polygons, for example, might contain a fill symbol layer to render the polygon interior, a stroke symbol layer for rendering the polygon outline, and a marker symbol layer to render polygon vertices. For a description of the types of symbol layers available for a multilayer symbol, see the symbol layers section of the symbol types described topic.

The following example creates a new multilayer polygon symbol with fill and stroke symbol layers and applies it to a polygon graphic.

```
// Create a blue solid fill symbol layer for the polygon interior.
FillSymbolLayer fillSymLayer = new SolidFillSymbolLayer(Color.Blue);

// Create a dark gray solid stroke symbol layer for the polygon outline.
StrokeSymbolLayer strokeSymLayer = new SolidStrokeSymbolLayer(8.0, Color.DarkGray);

// Add the symbol layers to a list and use it to create a multilayer polygon symbol.
List<SymbolLayer> symbolLayers = new List<SymbolLayer> { strokeSymLayer, fillSymLayer };
MultilayerPolygonSymbol multilayerPolySym = new MultilayerPolygonSymbol(symbolLayers);

// Apply the multilayer symbol to a polygon graphic.
Graphic parcelGraphic = new Graphic(parcelPolygon, multilayerPolySym);
```

Some symbol layers can contain additional symbol layers. A hatch fill symbol layer, for example, is composed of multilayer polyline symbols that define the hatch lines.

Note:
The order in which symbol layers are added dictates their drawing order in the symbol. The last layer in the collection will be the last layer to draw.

The following example uses a hatch fill symbol that contains a multilayer polyline symbol.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create two solid stroke layers to define the hatch: a wide blue line and a narrow white line.
SolidStrokeSymbolLayer wideHatchSymbolLayer = new SolidStrokeSymbolLayer(8, Color.DarkSlateBlue);
SolidStrokeSymbolLayer narrowHatchSymbolLayer = new SolidStrokeSymbolLayer(2, Color.WhiteSmoke);

// Create a multilayer polyline symbol from the stroke symbol layers.
List<SymbolLayer> hatchLineSymLayers = new List<SymbolLayer> { wideHatchSymbolLayer,
narrowHatchSymbolLayer };
MultilayerPolylineSymbol hatchLineSymbol = new MultilayerPolylineSymbol(hatchLineSymLayers);

// Create a hatch fill symbol layer that uses the polyline symbol to render hatches in the
polygon interior.
HatchFillSymbolLayer fillSymLayer = new HatchFillSymbolLayer(hatchLineSymbol, 45.0)
{
    // Set the hatch line separation (DIPs.
    Separation = 15
};

// Create a dark gray solid stroke symbol layer for the polygon outline.
StrokeSymbolLayer outlineSymLayer = new SolidStrokeSymbolLayer(8.0, Color.DarkGray);

// Add the symbol layers to a list and use it to create a multilayer polygon symbol.
List<SymbolLayer> symbolLayers = new List<SymbolLayer> { fillSymLayer, outlineSymLayer };
MultilayerPolygonSymbol multilayerPolySym = new MultilayerPolygonSymbol(symbolLayers);
```

Copyright © 1995-2020 Esri. All rights reserved.

Read symbols from a style file

For complex multilayer symbols, especially those you use frequently, it's easier to read them from a mobile style file (.stylx) than to create them from scratch. ArcGIS Pro provides an intuitive authoring environment for creating multilayer symbols. These symbols can be saved in a mobile style file (.stylx) and then consumed in your ArcGIS Runtime app.

Note:
Standard ArcGIS Pro style files and mobile styles both have a .stylx extension, but only mobile styles are supported in ArcGIS Runtime. Symbols from a standard style can be copied to a mobile style file, but may be generalized if they contain unsupported features, such as true curves or color gradients.

A mobile style can be opened in your app using the static `OpenAsync` method of `Esri.ArcGISRuntime.Symbology.SymbolStyle`. Each symbol in the mobile style has a unique key, name, tags, and category. You can reference a symbol using its key or search the style to return a set of results.

The following example opens a mobile style file and uses the default search parameters to return all symbols in the style.

```
 // Open a mobile style file.
SymbolStyle emojiStyle = await SymbolStyle.OpenAsync(_mobileStylesLocation +
"emoji-mobile.stylx");

// Get the default style search parameters.
SymbolStyleSearchParameters searchParams = await emojiStyle.GetDefaultSearchParametersAsync();

// Search the style with the default parameters to return all symbol results.
IList<SymbolStyleSearchResult> styleResults = await emojiStyle.SearchSymbolsAsync(searchParams);

// Loop through the results (perhaps look for symbols from a certain category).
foreach (SymbolStyleSearchResult result in styleResults)
{
   if(result.Category == "Hat")
   {
      hatSymsList.Add(result.Symbol);
   }
}
```

You can use a symbol's key to return it from the style. You can also get several symbols from a style as a single combined symbol. The `GetSymbolAsync` method of a `SymbolStyle` instance takes a list of identifiers (one or more keys) and returns a symbol that is a combination of all symbols that were found.

The following example uses a list of keys to create a symbol that contains four symbols from the mobile style.

```
// Open a mobile style file.
SymbolStyle emojiStyle = await SymbolStyle.OpenAsync(_mobileStylesLocation +
"emoji-mobile.stylx");

// Define the key for each symbol to combine.
List<string> keysList = new List<string>
{
    "Face1",
    "EyesWide",
    "MouthFrown",
    "PoliceHat"
};
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Get a symbol from the style that combines the identified symbols (in order).
Symbol faceSymbol = await emojiStyle.GetSymbolAsync(keysList);
```

Convert simple symbols to multilayer

The trade-off between simple and multilayer symbology is convenience versus control. Simple symbols are easy to create but offer a finite set of style options. Multilayer symbols often require a lot of code to create from scratch but give you fine-grained control and allow you to create more complex symbols. You might have simple symbols in your app that you'd like to convert to multilayer symbols, perhaps as a starting point for creating more complex types of symbols.

Simple marker, line, and fill symbols provide a `ToMultilayerSymbol` helper to convert to an equivalent multilayer symbol.

- Simple marker symbol is converted to a multilayer point symbol. A vector marker symbol element layer is added to the symbol layer collection to represent the original marker symbol (circle, square, triangle and so on). The vector marker symbol element layer geometry is a polygon.
- Simple line symbol is converted to a multilayer polyline symbol. A stroke symbol layer is added to the symbol layer collection to represent the original line symbol. If necessary, an effect is applied to the stroke layer to represent dash styles.
- Simple fill symbol is converted to a multilayer polygon symbol. A hatch or solid fill symbol layer is added to the symbol layer collection depending on the fill pattern (solid, horizontal, vertical and so on).

Once a simple symbol is represented as multilayer, you can add additional layers (perhaps from other simple symbols that you converted).

The following example converts a simple marker symbol and adds another symbol layer to draw hatch lines on the marker.

```
// Create a simple marker symbol (red circle).
SimpleMarkerSymbol simpleCircleMarker = new SimpleMarkerSymbol(SimpleMarkerSymbolStyle.Circle,
Color.Red, 20);

// Convert the simple marker to a multilayer point symbol.
MultilayerPointSymbol multilayerPointSymbol = simpleCircleMarker.ToMultilayerSymbol();

// Get the first (only) symbol layer from the new multilayer point symbol.
VectorMarkerSymbolLayer vectorMarkerSymLayer =
multilayerPointSymbol.SymbolLayers.FirstOrDefault() as VectorMarkerSymbolLayer;

// Get the first (only) element in the vector marker symbol layer (it's a circle polygon
element).
VectorMarkerSymbolElement circleVectorMarkerSymElement =
vectorMarkerSymLayer.VectorMarkerSymbolElements.FirstOrDefault();

// Create a stroke symbol layer for hatch lines (white solid stroke).
SolidStrokeSymbolLayer hatchStrokeLayer = new SolidStrokeSymbolLayer(1, Color.White);

// Use the stroke layer to create a new multilayer polyline symbol for the hatch lines.
MultilayerPolylineSymbol multilayerPolylineSym = new MultilayerPolylineSymbol(new
List<SymbolLayer> { hatchStrokeLayer });
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create a new hatch fill symbol layer with the polyline symbol. Specify a line separation of 8
DIPs.
HatchFillSymbolLayer hatchFillLayer = new HatchFillSymbolLayer(multilayerPolylineSym, 45)
{
    Separation = 8
};

// Create a new multilayer polygon symbol with the hatch fill layer.
MultilayerPolygonSymbol multilayerPolygonSym = new MultilayerPolygonSymbol(new List<SymbolLayer>
{ hatchFillLayer });

// Get the geometry from the vector marker symbol element (polygon).
Geometry vectorElementGeometry = circleVectorMarkerSymElement.Geometry;

// Create a new vector marker symbol element to display the hatch fill, use the same marker
shape (circle polygon) for the geometry.
VectorMarkerSymbolElement hatchVectorMarkerSymElement = new
VectorMarkerSymbolElement(vectorElementGeometry, multilayerPolygonSym);

// Update the multilayer point symbol by adding the new hatch element to the vector marker
symbol layer.
vectorMarkerSymLayer.VectorMarkerSymbolElements.Add(hatchVectorMarkerSymElement);

// Apply the symbol to a point graphic.
Graphic pointGraphic = new Graphic(tapPoint, multilayerPointSymbol);
```

Create 3D symbols

While you can use the same 2D symbols available for symbolizing graphics in a map view, ArcGIS Runtime provides some specialized symbols for displaying graphics in a scene view. You can display points and lines as 3D shapes or use models to realistically represent real-world objects (such as vehicles). See 3D-specific symbols for more information.

Copyright © 1995-2020 Esri. All rights reserved.

Add shape-based symbols

The following code snippet shows how to create a sphere symbol and render it using a point geometry in a graphic.

```
// create a red sphere symbol
var sphereSymbol = new SimpleMarkerSceneSymbol
{
    Style = SimpleMarkerSceneSymbolStyle.Sphere,
    Color = Color.Red,
    Height = 3000,
    Width = 3000,
    Depth = 3000,
    AnchorPosition = SceneSymbolAnchorPosition.Center
};

// create a graphic using a map location (point) and the sphere symbol
var location = new MapPoint(-4.04, 53.16, 1000, SpatialReferences.Wgs84);
var sphereGraphic = new Graphic(location, sphereSymbol);

// add the graphic to the first graphics overlay in the scene view
MySceneView.GraphicsOverlays[0].Graphics.Add(sphereGraphic);
```

Add model symbols

Create these model symbols using the `ModelSceneSymbol` class.

```
// call the static CreateAsync method to create a new ModelMarkerSymbol, pass a uri for the
model and a scale
var modelSymbol = await ModelSceneSymbol.CreateAsync(modelSourceUri, 300.0);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// create a graphic using a map location (point) and the model symbol
var location = new MapPoint(-4.04, 53.16, 2000, SpatialReferences.Wgs84);
var modelGraphic = new Graphic(location, modelSymbol);

// add the graphic to the first graphics overlay in the scene view
MySceneView.GraphicsOverlays[0].Graphics.Add(modelGraphic);
```

Distance composite symbols

While model symbols closely resemble the real-world object they represent when viewed close up, when you zoom away from them, the symbols gets smaller until they're no longer visible. If you want to see the location of your graphics at any distance from the camera, use distance composite symbols. These symbols allow you to use different symbols depending on the distance the graphic is from the camera.

The code snippet below the following images shows how to set up a distance composite symbol similar to the one shown in the images.

When the camera is far away from the point, a red simple marker symbol displays.

As you zoom closer to the point, the symbol renders as a cone pointing in the aircraft's direction of travel.

When viewed even closer, the point displays as a model symbol, which is appropriate at this distance from the camera.

Copyright © 1995-2020 Esri. All rights reserved.

```
// create a new DistanceCompositeSceneSymbol to hold a collection of range-specific symbols
var compositeSymbol = new DistanceCompositeSceneSymbol();

// ... code here to create new ModelMarkerSymbol, ConeMarkerSymbol, and SimpleMarkerSymbol ...

// use a model symbol when the camera is within 5,000 meters
var nearSymbolInfo = new DistanceSymbolRange(modelSymbol, 0, 5000);
compositeSymbol.Ranges.Add(nearSymbolInfo);

// use a cone marker symbol for intermediate distances (5,000 to 50,000 meters)
var midSymbolInfo = new DistanceSymbolRange(coneSymbol, 5000, 50000);
compositeSymbol.Ranges.Add(midSymbolInfo);

// use a simple marker symbol for large distances (> 50,000 meters)
var farSymbolInfo = new DistanceSymbolRange(markerSymbol, 50000, 0);
compositeSymbol.Ranges.Add(farSymbolInfo);

// create a new graphic using the DistanceCompositeSceneSymbol
var location = new MapPoint(-4.04, 53.16, 2000, SpatialReferences.Wgs84);
var distanceCompositeGraphic = new Graphic(location, compositeSymbol);

// add the graphic to the first graphics overlay in the scene view
MySceneView.GraphicsOverlays[0].Graphics.Add(distanceCompositeGraphic);
```

Apply a renderer

A renderer is an object that determines how geoelements (features or graphics) are displayed. A renderer includes one or more symbols as well as logic to determine which symbol to apply for data in the layer (or graphics overlay). There are a variety of renderer types, each designed to use a different rendering logic. Renderers use attribute values to determine which symbol should be applied.

The Symbols, renderers, and styles topic provides an overview of symbols and renderers and how they work together to symbolize data.

Note:

If you apply a symbol to a graphic that is in a graphics overlay with a renderer, the symbol on the graphic will take precedence over the renderer.

Copyright © 1995-2020 Esri. All rights reserved.

Much of the data you bring into your ArcGIS Runtime app includes default rendering information. Layers from a web map, service, or mobile map package, for example, are displayed using the symbols applied when they were authored. You can make changes to an existing renderer for a layer, or create and apply new renderers. Be aware that some rendering behavior, while honored by your app, cannot be modified using ArcGIS Runtime SDK. Unique value renderers and class breaks renderers, for example, can be authored using ArcGIS Pro to define scale ranges for applying associated symbols. These scale ranges will be used when displaying the layer in an ArcGIS Runtime app but no API is exposed for creating or modifying ranges. See Scale-based symbol classes in the ArcGIS Pro documentation for more information about using scale ranges for these renderers.

ArcGIS Runtime SDK provides simple, unique value, class breaks, and dictionary renderers for displaying geoelements.

The following steps describe the process for defining a renderer for a feature layer or graphics overlay. Dictionary renderers are described in the Display military symbols with a dictionary renderer topic. See the Raster renderers section for examples of changing raster display.

1. Create a new renderer of the desired type: simple, unique value, or class breaks.
2. Create the symbol(s) required by the renderer and appropriate for the geometry type. The Symbol types described topic provides an overview of the available ArcGIS Runtime symbols.
3. Define the logic that the renderer needs in order to apply the correct symbol for each geoelement. Depending on the type of renderer, this could be one or several attribute names or ranges of values for a given numeric attribute. A simple renderer always applies a single symbol to all geoelements.
4. For unique value or class breaks renderers, associate the desired symbol for each range or unique value defined in the renderer.
5. Apply the renderer by assigning it to the to the feature layer or graphics overlay.

Simple renderer

A simple renderer uses one symbol for all features in a layer (or all graphics in an overlay) regardless of attribute values.

Unique value renderer

A unique value renderer symbolizes features in a layer (or graphics in an overlay) differently based one or more of their attribute values. It's useful for symbolizing features based on an attribute whose values represent names or categories, usually referred to as **nominal data**. Unlike graphics, features do not have a symbol property and therefore cannot be assigned an individual symbol. However, you can use a unique value renderer to set symbology on a layer based on each feature's attribute values. (You can do this for graphics, too).

Note:
ArcGIS map services allow up to three fields to be used for unique value renderers. An ArcGIS feature service allows only one. As a developer, you can specify as many fields as you like, ensuring that the order of the fields you specify corresponds to the order of the values you specify.

The example below creates a unique value renderer that displays US states with different symbols according to the value of their state abbreviation attribute. Since only California, Nevada, and Arizona have been defined explicitly in the renderer's `UniqueValueInfoCollection`, features with any other value for the state abbreviation attribute (`STATE_ABBR`) display with the default (transparent) symbol.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
Features whose attribute value falls outside the classes defined for the renderer are symbolized with the default symbol. If a default symbol is not defined, those features won't display.

Class breaks renderer

Use a class breaks renderer to symbolize quantitative information about feature data. Class breaks symbology divides the range of feature values into a relatively small number of classes. For example, you can use a class breaks renderer to symbolize geoelements by population value, dividing the features into three classes of low, medium, and high population value.

One symbol is associated with each class, and is used to render geoelements whose attribute value falls within the range defining the class. For example, with polygons, you could use a yellow fill symbol to represent low population, an orange fill for medium population, and red fill for high population. If a geoelement has an attribute value that falls outside all class ranges, a default symbol can be applied (or no symbol).

Symbol color is an effective way to represent differences in magnitude of a phenomenon because it is easy to distinguish variations in color if there are relatively few classes. A range of seven colors is the approximate upper limit of colors that can be easily distinguished on a map. Avoid using too many classes, especially if you are using light colors.

A class breaks renderer can be used for any geometry type and the symbols do not have to only vary by color. For example, a line feature class representing roads with associated numeric values for traffic density can be rendered using different line widths for high, medium, and low traffic. City points can be classified with a range of circular marker symbols with different sizes or colors (or both) based on population.

```
// Create a new service feature table using the USA States url.
ServiceFeatureTable statesFeatureTable = new ServiceFeatureTable(new
Uri("https://sampleserver6.arcgisonline.com/arcgis/rest/services/Census/MapServer/3"));

// Create a new feature layer using the service feature table.
FeatureLayer statesLayer = new FeatureLayer(statesFeatureTable);

// Create a new class breaks renderer.
ClassBreaksRenderer statePopRenderer = new ClassBreaksRenderer
{
    // Define "POP2007" as the field containing the values for the class breaks renderer.
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    FieldName = "POP2007"
};

// Create an outline symbol to use for each polygon.
SimpleLineSymbol outlineSymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Solid,
Color.LightGray, 1);

// Define the symbols to use for each population range.
SimpleFillSymbol popSymbol1 = new SimpleFillSymbol(SimpleFillSymbolStyle.Solid,
Color.FromArgb(150, Color.Yellow), outlineSymbol);
SimpleFillSymbol popSymbol2 = new SimpleFillSymbol(SimpleFillSymbolStyle.Solid,
Color.FromArgb(150, Color.Orange), outlineSymbol);
SimpleFillSymbol popSymbol3 = new SimpleFillSymbol(SimpleFillSymbolStyle.Solid,
Color.FromArgb(150, Color.Red), outlineSymbol);

// Define the ranges of values to render with (class breaks).
ClassBreak popBreak1 = new ClassBreak("Low", "0 - 1M", 0, 999999, popSymbol1);
ClassBreak popBreak2 = new ClassBreak("Medium", "1M - 10M", 1000000, 9999999, popSymbol2);
ClassBreak popBreak3 = new ClassBreak("High", "10M - 40M", 10000000, 40000000, popSymbol3);

// Add the class breaks to the renderer.
statePopRenderer.ClassBreaks.Add(popBreak1);
statePopRenderer.ClassBreaks.Add(popBreak2);
statePopRenderer.ClassBreaks.Add(popBreak3);

// Set the default fill symbol (transparent with no outline) for any population values not
within a range defined in the renderer.
SimpleFillSymbol defaultFillSymbol = new SimpleFillSymbol(SimpleFillSymbolStyle.Null,
Color.Transparent, null);
statePopRenderer.DefaultSymbol = defaultFillSymbol;
statePopRenderer.DefaultLabel = "Other";

// Apply the class breaks renderer to the states layer.
statesLayer.Renderer = statePopRenderer;

// Add the new layer to the map.
MyMapView.Map.OperationalLayers.Add(statesLayer);
```

The following image shows a class breaks renderer with three class breaks. The US States layer is classified by three ranges of population for the year 2007.

Copyright © 1995-2020 Esri. All rights reserved.

3D renderer properties

When using renderers in 3D, you can set properties to define extrusion expressions for both features and graphics. For graphics you can also define a rotation expression to control heading, pitch, and roll.

Extrusion is the process of stretching a flat 2D shape vertically to create a 3D object. This provides a simple method to create three-dimensional symbology from two-dimensional features. For example, you can extrude building polygons by a height value to create realistic building shapes. You can also use extrusion to illustrate relative values in your data. The three basic geometry types-points, lines, and polygons-all support extrusion.

Note:
Extrusion may not work with some types of multilayer symbols, such as hatch fills.

To apply extrusion to a feature layer:

1. Set the feature layer to rendering in `DYNAMIC` mode
2. Define a renderer which will be used to render the feature layer
3. Set the renderer's scene property extrusion mode
4. Set the renderer's scene property extrusion expression
5. Set the renderer to the feature layer
6. Add the feature layer to the scene

Extrusion can also be applied to graphics overlays:

1. Define a renderer which will be used to render the graphics overlay
2. Set the renderer's scene property extrusion mode
3. Set the renderer's scene property extrusion expression
4. Set the renderer to the graphics overlay

```
// create a simple renderer that uses a (polygon) fill symbol
var lineSym = new SimpleLineSymbol(SimpleLineSymbolStyle.Solid, Color.Black, 1);
var fillSym = new SimpleFillSymbol(SimpleFillSymbolStyle.Solid, Color.Green, lineSym);
var rendrr = new SimpleRenderer(fillSym);

// set the rendering mode (absolute) and the expression (population attribute name)
rendrr.SceneProperties.ExtrusionMode = ExtrusionMode.AbsoluteHeight;
rendrr.SceneProperties.ExtrusionExpression = "pop2000";

// apply the renderer to the first graphics overlay in the scene view
MySceneView.GraphicsOverlays[0].Renderer = rendrr;
```

The following image shows a graphic overlay of US counties extruded based on a population attribute.

Copyright © 1995-2020 Esri. All rights reserved.

Only graphics overlays can have rotation expressions--used to change the heading, pitch and roll of the graphics in the graphics overlay. To set rotation expressions:

1. Define a renderer which will be used to render the graphics overlay
2. Set the rotation type
3. Set the renderer's scene property heading expression
4. Set the renderer's scene property pitch expression
5. Set the renderer's scene property roll expression
6. Set the renderer to the graphics overlay

```
// create a simple renderer that uses a (3D) cone symbol
var rendrr = new SimpleRenderer(coneSymbol);
// set expressions (attribute names) to define heading, pitch, and roll for each graphic
rendrr.SceneProperties.HeadingExpression = "heading";
rendrr.SceneProperties.PitchExpression = "pitch";
rendrr.SceneProperties.RollExpression = "roll";
```

Raster renderers

When initially created, a default renderer is used to display a raster layer. To customize the display, you can create a raster renderer (`RasterRenderer`) to define display properties and apply it to the layer.

The following code defines a color ramp and stretch parameters to create a stretch raster renderer.

```
double[] minVals = { 0 };
double[] maxVals = { 100 };
MinMaxStretchParameters minMaxParams = new MinMaxStretchParameters(minVals, maxVals);
ColorRamp elevationColorRamp = ColorRamp.Create(PresetColorRampType.Elevation);
var rendrr = new StretchRenderer(minMaxParams, null, true, elevationColorRamp);
inRasterLayer.Renderer = rendrr;
```

Copyright © 1995-2020 Esri. All rights reserved.

`RasterRenderer` is the base class inherited by all renderers available to display a raster layer and allows you to control how raster data are presented (visualized). Each of the following renderers are types of `RasterRenderer` and can be used to display your raster layer in different ways.

- `HillshadeRenderer`—Creates a grayscale 3D representation of an elevation surface, with the sun's (hypothetical) position taken into account for shading the image. It can be applied to a raster layer created with single-band raster data.
- `BlendRenderer`—Blends a hillshade image (derived from the raster) with the original raster. This provides a look similar to the original raster but with some terrain shading, for a rich, textured look.
- `ColormapRenderer`—Provides a discrete mapping of raster pixel values to colors. All pixels matching the specified value are rendered using the mapped color. This can be useful for tasks such as land classification.
- `StretchRenderer`—Displays continuous raster cell values across a gradual ramp of colors. Use the stretch renderer to draw a single band of continuous data. The stretch renderer works well when you have a large range of values to display, such as in imagery, aerial photographs, or elevation models.
- `RGBRenderer`—Uses the same methods as the stretch renderer but allows you to combine bands as red, green, and blue composites.

Raster functions can also be used to apply on-the-fly processing to rasters to apply hillshade, colormaps, color ramps, and so on. See the Add raster data topic for more information.

Raster rendering rules

A rendering rule defines how a requested raster image should be rendered or processed. `ImageServiceRaster` allows you to apply service-defined or client-defined rendering rules on an image service by setting its rendering rule property. You can

Copyright © 1995-2020 Esri. All rights reserved.

create a `RenderingRule` from a `RenderingRuleInfo` object or a JSON string that specifies a service-defined or client-defined rendering rule and then set it on the image service raster. For the syntax of rendering rules, see raster function (JSON) objects in the ArcGIS REST API help.

```
private void AddRasterWithRenderingRule(RenderingRuleInfo ruleInfo)
{
    // Create an image service raster from a URI
    ImageServiceRaster serviceRaster = new ImageServiceRaster(new
Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/CharlotteLAS/ImageServer"));

    // Use RenderingRuleInfo to create a rendering rule
    RenderingRule renderingRule = new RenderingRule(ruleInfo);

    // Apply the rendering rule to the image service raster
    serviceRaster.RenderingRule = renderingRule;

    // Create a raster layer to display the raster
    RasterLayer layer = new RasterLayer(serviceRaster);

    // Dispatch to the UI thread to show the raster layer as a basemap
    Dispatcher.Invoke(() =>
    {
        MyMapView.Map = new Map(new Basemap(layer));
    });
}
```

Alternatively you can create a `RenderingRule` using a JSON string that specifies a service-defined or client-defined rendering rule.

```
// Use a JSON string to define the RenderingRule
RenderingRule renderingRule = new RenderingRule("{\"rasterFunction\" :
\"Hillshade\",\"rasterFunctionArguments\" : {\"Azimuth\" : 215.0,\"Altitude\" : 75.0,\"ZFactor\"
: 0.3},\"variableName\" : \"DEM\"}");
```

Symbolize graphics

Graphics provide a basic way to display geometry on top of the map. Graphics are unique in that they can be symbolized using a renderer for the overlay that contains them or by applying a symbol directly to the graphic. For a description of how to work with graphics, see Add graphics and text to graphics overlays. To apply symbols to features (as opposed to graphics), create a renderer.

The following example shows how to create a point graphic, apply a symbol, then display it on the map.

Note:
A symbol applied directly to a graphic overrides the symbol applied through the graphics overlay's renderer.

Apply styles to layers

Some layers that don't support symbols and renderers, such ArcGIS vector tiled layers and WMS, offer styles as a way to control the display of the features they contain. These layers use a default style and also provide the option to apply other available styles.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
The styles described here are not the styles you can create using ArcGIS Pro (.stylx). For information about reading symbols from such a style file, see Read symbols from a map or style file.

ArcGIS vector tiled layer styles

With ArcGIS Runtime SDK you can display vector basemaps with your own custom style. The ArcGIS Vector Tile Style Editor allows you to interactively create a style for one of the Esri vector basemaps and save it in ArcGIS Online as a portal item. You can then open it from the portal item to display it in your map with the custom style.

Hands-on instructions for creating a custom style and displaying it in an ArcGIS Runtime app are in the Display a styled vector basemap DevLab.

WMS styles

WMS servers provide clients with a list of supported styles for each layer. At run time, you can choose which style the WMS server uses to render map images. In general, styles are predefined and cannot be changed or added to.

The `styles` property of the `WmsLayerInfo` class can be inspected to determine if there are styles available. The current style property of a WMS sublayer can be set to select a specific style.

```
// Search for and apply a specific style on sublayers that support it
foreach(WmsSublayer sublayer in _myWMSLayer.Sublayers)
{
    if (sublayer.SubLayerInfo.Styles.Contains("ArcGIS"))
    {
        // Set the style
        sublayer.CurrentStyle = "ArcGIS";
    }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Add labels

Features and graphics in your ArcGIS Runtime SDK app can be labeled using a combination of attribute values, text strings, and values calculated with an expression. You can determine how labels are positioned and prioritized, and how conflicts between overlapping labels are automatically and dynamically resolved. Define any number of label classes for a layer to set unique labels for distinct groups of features.

Note:
Label classes on Feature Layers in ArcGIS Pro are used to generate dynamic labels (which resize and move with changes in the map view). Should the position and size of text need to be fixed, you can convert labels to static annotation features in an Annotation layer using ArcGIS Pro.

Define labels for features or graphics

Feature layers and graphics overlays provide the ability to show or hide labels in the map and to control a variety of labeling behavior. A **label definition** is used to specify:

- what labels look like (font, size, color, angle, and so on)
- at which scales they should be visible
- the text to display (which attributes, for example)
- which features should be labeled
- how to handle overlaps and prioritize labels

If you want to label all features in your layer the same way, you can define labeling with a single label definition. If you want to display different labels for different types of features, you can add as many label definitions as you need to define distinct sets of features for labeling. For graphics overlays and feature layers, labels are rendered by the client app.

Note:
At this release, labeling of feature layers and graphics overlays is only supported in 2D (maps). In 3D (scenes), only map image sublayer labeling is supported.

A label definition is constructed as a JSON string using the syntax defined in the Web map specification for label definition. In order to provide a more accurate representation of labels created in ArcGIS Pro (for a mobile map package, for example), ArcGIS Runtime defines additional JSON properties that are not included in the Web map specification. Refer to the JSON label class properties topic for a description of the available properties, their expected values, and the defaults used for each.

Most properties in the label definition are optional and will use a default value if not assigned explicitly. At a minimum, you must provide a `labelExpressionInfo` (or `labelExpression`), a `labelPlacement` option, and a `symbol` (with at least color, font size, and type) to display labels for a layer. The JSON below provides the most basic label definition for a layer.

```
{
  "labelExpressionInfo":
  {
    "expression": "return $feature.address;"
  },
  "labelPlacement": "esriServerPolygonPlacementAlwaysHorizontal",
  "symbol":
  {
    "color": [255,0,255,123],
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    "font": { "size": 16 },
    "type": "esriTS"
  }
}
```

The preceding example uses an Arcade expression for the labelExpressionInfo to label each feature with the value of its `address` attribute. See the Arcade documentation for information about working with Arcade expressions across the ArcGIS platform. The use of Arcade expressions is recommended when labeling feature layers or graphics overlays.

Caution:

Label definitions for a map image layer must use the ArcGIS Server REST API for label expressions. Arcade expressions are not supported for labels rendered by ArcGIS Server.

In your ArcGIS Runtime SDK code, the JSON label definitions are managed by the `LabelDefinition` class. Two static methods on the class, `FromJson` and `ToJson`, allow you to serialize and deserialize JSON label definitions. Feature layers, map image sublayers, and graphics overlays all maintain a collection of label definitions in a `LabelDefinitions` property. Labels for a layer or graphics overlay can be shown or hidden using the `LabelsEnabled` property.

Note:

ArcGIS Runtime will render labels for web map layers that have label definitions stored with them. You can use the `LabelDefinition.ToJson` method to deserialize label definitions stored with a web map to a JSON string.

The following example creates a label definition and adds it to a feature layer's label definition collection.

```
// Create a StringBuilder to create the label definition JSON string
StringBuilder addressLabelsBuilder = new StringBuilder();
addressLabelsBuilder.AppendLine("{");
//     Define a labeling expression that will show the address attribute value
addressLabelsBuilder.AppendLine("\"labelExpressionInfo\": {");
addressLabelsBuilder.AppendLine("\"expression\": \"return $feature.address;\"},");
//     Align labels horizontally
addressLabelsBuilder.AppendLine("\"labelPlacement\":
\"esriServerPolygonPlacementAlwaysHorizontal\",");
//     Use a green bold text symbol
addressLabelsBuilder.AppendLine("\"symbol\": {");
addressLabelsBuilder.AppendLine("\"color\": [0,255,50,255],");
addressLabelsBuilder.AppendLine("\"font\": {\"size\": 18, \"weight\": \"bold\"},");
addressLabelsBuilder.AppendLine("\"type\": \"esriTS\"}");
addressLabelsBuilder.AppendLine("}");

// Get the label definition string
var addressLabelsJson = addressLabelsBuilder.ToString();

// Create a new LabelDefintion object using the static FromJson method
LabelDefinition labelDef = LabelDefinition.FromJson(addressLabelsJson);

// Clear the current collection of label definitions (if any)
_parcelsLayer.LabelDefinitions.Clear();

// Add this label definition to the collection
_parcelsLayer.LabelDefinitions.Add(labelDef);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Make sure labeling is enabled for the layer
_parcelsLayer.LabelsEnabled = true;
```

Common aspects of label behavior that you can control for a label class are described in the following section (JSON property names in parentheses). For a comprehensive list that includes expected values, defaults, and some examples see the JSON label class properties topic.

- **Label text** (`labelExpressionInfo.expression`)—A label expression can be used to determine the text to display for each feature in the label definition. The label text can come from a combination of available attributes, text strings, and expressions. A label showing length in meters from values in feet from an attribute named `length_ft`, for example, may look like this: `($feature.length_ft * 0.3048) + ' meters'`. A feature with a `length_ft` value of `343` would result in a label of `104.546 meters`.
- **Text symbol** (`symbol`)—The font, size, color, angle, and so on, used to display labels in the definition. You can also provide a border or background for the label symbol.
- **Maximum scale** (`maxScale`)—The largest scale at which labels in the definition are shown. This value represents the scale denominator for the zoomed-in scale; a value of 0 means a maximum scale is not applied.
- **Minimum scale** (`minScale`)—The smallest scale at which labels in the definition are shown. This value represents the scale denominator for the zoomed-out scale; a value of 0 means a minimum scale is not applied.
- **Placement** (`labelPlacement`)—Labels can be placed at a specified position relative to the features they describe. There are different options for placement depending on the geometry of the feature being labeled. Line features, for example, may have labels placed above the center of the line, below the center of the line, above the end point, and so on.
- **Position** (`deconflictionStrategy`)—Logic for positioning labels can be controlled using a variety of options, such as whether they should be allowed to overlap with features in the layer.
- **Priority** (`priority`)—A label definition can be given a priority relative to other definitions in the layer. If labels from different definitions conflict in their placement, the label from the highest-priority definition will be displayed.
- **Label overlap with other labels** (`allowOverlapOfLabel`)—Whether other labels are allowed to overlap this label. One of the following values is expected: `allow` means that labels are allowed to overlap this label, `avoid` means that labels that

Copyright © 1995-2020 Esri. All rights reserved.

would overlap will move as much possible to minimize the overlap, and `exclude` (the default) means that labels that would overlap are not placed.

- **Label overlap with features**—Whether other labels are allowed to overlap a polygon edge (`allowOverlapOfFeatureBoundary`) or feature (`allowOverlapOfFeatureInterior`). One of the following values is expected: `allow` means that labels are allowed to overlap, `avoid` means that labels that would overlap will move as much possible to minimize the overlap, and `exclude` means that labels that would overlap a feature are not placed. The default value is `allow` for lines and polygons and `exclude` for points.
- **Which features to label** (`where`)—The features labeled with the label definition are determined by evaluating an attribute expression. If this expression is not defined, all features are included in the definition. When using multiple label definitions, it's important that expressions uniquely assign features for each definition.

Creating multiple label definitions for a single layer is useful when you want to distinguish labels for certain types of features. When labeling a cities layer, for example, you may want to display capital cities with a larger font or different color.

Use text formatting tags with labels

Text used for labels can contain some of the same formatting tags used by ArcGIS Pro. These tags are similar to html tags and are used to modify the appearance of the text. For the current release, ArcGIS Runtime supports the following tags:

- `<FNT>`—Font name, size, and scaling.
- `<CLR>`—Color (defined with RGB or CMYK values).
- `<BOL>`, `<_BOL>`—Bold / un-bold.
- `<ITA>`, `<_ITA>`—Italic / un-italic.
- `<UND>`, `<_UND>`—Underline / un-underline.

See the Text formatting tags topic in the ArcGIS Pro documentation for more information about these formatting tags and how they can be used with your labels.

Define labels for a map image layer

Labels for map image sublayers (`ArcGISMapImageLayer`) can be defined in much the same way as labels for feature layers and graphics overlays. The main difference is that labels for map image layers are rendered by the server and not in your client app. You must therefore use ArcGIS Server REST API syntax to define your label expressions, rather than the Arcade syntax you can use with feature layers and graphics overlays. Labels for a map image sublayer are supported for layers displayed in 2D (maps) and in 3D (scenes).

The JSON below creates a label definition that uses `labelExpression` with ArcGIS Server REST syntax. This differs from the preceding example for graphics and features that used `labelExpressionInfo.expression` and Arcade syntax.

```
{
  "labelExpression": "[areaname]",
  "labelPlacement": "esriServerPointLabelPlacementAboveCenter",
  "symbol": {
    "color": [255,0,255,123],
    "font": {"size": 16},
    "type": "esriTS"
  }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

The following example creates two label definitions (with REST syntax for the label expressions) and adds them to the label definitions collection for a map image sublayer. The `where` expression is used to identify features for each label definition, one for capital cities and one for all other cities.

```
// Define two label definitions: one for capital cities and one for everything else
StringBuilder capitalLabelsBuilder = new StringBuilder();
StringBuilder otherLabelsBuilder = new StringBuilder();

// Use a StringBuilder to create the label definition JSON string for capital cities
capitalLabelsBuilder.AppendLine("{");
//     Define a labeling expression (REST syntax) that will show the 'areaname' attribute value
capitalLabelsBuilder.AppendLine("\"labelExpression\": \"[areaname]\",");
//     Align labels above the center of each point
capitalLabelsBuilder.AppendLine("\"labelPlacement\":
\"esriServerPointLabelPlacementAboveCenter\",");
//     Use a white bold text symbol
capitalLabelsBuilder.AppendLine("\"symbol\": {");
capitalLabelsBuilder.AppendLine("\"color\": [255,255,255,255],");
capitalLabelsBuilder.AppendLine("\"font\": {\"size\": 18, \"weight\": \"bold\"},");
capitalLabelsBuilder.AppendLine("\"type\": \"esriTS\"},");
//     This definition is for cities with a value of 'Y' for the capital attribute
capitalLabelsBuilder.AppendLine("\"where\": \"[capital] = 'Y'\"");
capitalLabelsBuilder.AppendLine("}");

// Get the label definition string for capital cities
var capitalLabelsJson = capitalLabelsBuilder.ToString();

// Create a LabelDefintion object for capital cities using the static FromJson method
LabelDefinition capitalsLabelDef = LabelDefinition.FromJson(capitalLabelsJson);

// Use a StringBuilder to create the label definition JSON string for non-capitals
otherLabelsBuilder.AppendLine("{");
//     Define a labeling expression (REST syntax) that will show the 'areaname' attribute value
otherLabelsBuilder.AppendLine("\"labelExpression\": \"[areaname]\",");
//     Align labels above the center of each point
otherLabelsBuilder.AppendLine("\"labelPlacement\":
\"esriServerPointLabelPlacementAboveCenter\",");
//     Use a smaller beige text symbol
otherLabelsBuilder.AppendLine("\"symbol\": {");
otherLabelsBuilder.AppendLine("\"color\": [255,255,110,255],");
otherLabelsBuilder.AppendLine("\"font\": {\"size\": 12},");
otherLabelsBuilder.AppendLine("\"type\": \"esriTS\"},");
//     This definition is for cities with a value of 'N' for the capital attribute
otherLabelsBuilder.AppendLine("\"where\": \"[capital] = 'N'\"");
otherLabelsBuilder.AppendLine("}");

// Get the label definition string for non-capitals
var otherLabelsJson = otherLabelsBuilder.ToString();

// Create a LabelDefintion object for non-capitals using the static FromJson method
LabelDefinition othersLabelDef = LabelDefinition.FromJson(otherLabelsJson);

// Get the cities sublayer (first sublayer in an ArcGISMapImageLayer)
ArcGISMapImageSublayer subLayer = _citiesLayer.Sublayers[0] as ArcGISMapImageSublayer;

// Enable labels
subLayer.LabelsEnabled = true;

// Clear any existing label definitions
subLayer.LabelDefinitions.Clear();

// Add the two label definitions to the collection
```

Copyright © 1995-2020 Esri. All rights reserved.

```
subLayer.LabelDefinitions.Add(capitalsLabelDef);
subLayer.LabelDefinitions.Add(othersLabelDef);
```

Note:
The label definitions in the preceding code would also work for a feature layer or graphics overlay (assuming the 'areaname' attribute is present in the data).

Copyright © 1995-2020 Esri. All rights reserved.

Add raster data

You can add raster data stored on your desktop or device or from an image service to your map or scene by doing the following:

- Adding a raster using a raster layer if you want to display the raster
- Adding a raster for use with a raster function

Note:
Options for customizing raster a layer display are described in the Symbolize data topic.

Raster data provides a unique way of analyzing and visualizing geographic phenomena. Rasters consist of a matrix of cells, with each cell containing a value. Rasters can have multiple dimensions or bands, for example, the red, green, and blue channels in an aerial photo. Rasters are different from vectors, which use points, lines, and polygons to represent features. For more information on raster data, including advantages and disadvantages of using it, see What is raster data? in the ArcGIS Desktop help.

Raster data is represented by the `Raster` class. Your ArcGIS Runtime app can work with many sources of raster data including raster files, mosaic datasets, raster data shared through image services, or the results of raster functions. A raster function can take any of these rasters as input for analysis.

The `Raster` object implements the loadable pattern, which asynchronously accesses raster data from remote services or datasets.

Raster data supports the identify operation to obtain `RasterCell` values for pixels clicked on via user interaction in a `MapView` or `SceneView` for `RasterLayers`.

Create a raster from a raster file

To add raster files stored on your desktop or device to your app, create a`Raster` object using the path of the raster file.

```
// create a raster from a path to a supported raster format
Raster myRaster = new Raster(pathToRaster);

// create a RasterLayer using the Raster
RasterLayer newRasterLayer = new RasterLayer(myRaster);
```

Load a raster from a GeoPackage

A **GeoPackage** is an open, standards-based, platform-independent, portable, self-describing, compact format for transferring geospatial information. It is a platform-independent SQLite database file that contains data and metadata tables. For details, refer to the OGC GeoPackage specification. Using ArcGIS Runtime SDK, you can read a local GeoPackage and load the feature tables and rasters it contains.

Open a `GeoPackage` using the path to the file (`.gpkg`). The `GeoPackageRasters` property returns a collection of all rasters in the package. You can work with a `GeoPackageRaster` as you would any other `Raster`, including adding it as a `RasterLayer` to your app's map.

```
// Open a local GeoPackage (.gpkg)
GeoPackage myGeoPackage = await GeoPackage.OpenAsync(geoPackageFilePath);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Get the collection of rasters in the package
IReadOnlyList<GeoPackageRaster> packageRasters = myGeoPackage.GeoPackageRasters;

// Get the first raster in the collection
GeoPackageRaster firstRaster = packageRasters[0];

// Create a RasterLayer using the GeoPackageRaster (which inherits from Raster)
RasterLayer newRasterLayer = new RasterLayer(firstRaster);
```

Create a raster from a mobile mosaic dataset

Raster data stored in a mobile mosaic dataset should be read through the `MosaicDatasetRaster`, which inherits from `Raster`. The `MosaicDatasetRaster` object is instantiated with a path of the mobile geodatabase and a name of the dataset.

Note:
You can create a mobile mosaic dataset by using the Mosaic Dataset to Mobile Mosaic Dataset geoprocessing tool in ArcGIS Pro.

```
// Get mosaic dataset names in the SQLite database.
var names = MosaicDatasetRaster.GetNames(sqliteDbPath);
var rasterName = names[0];

// Create a raster from a mosaic dataset
MosaicDatasetRaster raster = new MosaicDatasetRaster(sqliteDbPath, rasterName);

// Create a RasterLayer to display the Raster
RasterLayer rasterLayer = new RasterLayer(raster);
```

Create a mobile mosaic dataset and add raster files

`MosaicDatasetRaster` can also be used to create a mobile mosaic dataset in a new mobile geodatabase and save it to the device. Once the mobile mosaic dataset is created, individual raster files can be asynchronously added to it. The mobile geodatabase is created when the `MosaicDatasetRaster` is loaded successfully.

```
// Create a new mobile mosaic dataset.
MosaicDatasetRaster rasterMosaic = MosaicDatasetRaster.Create(@"C:\Data\mosaic.sqlite",
"Shasta", SpatialReferences.WebMercator);

// When the new mosaic dataset is ready, add some raster files.
rasterMosaic.LoadStatusChanged += (s,e)=>
{
    // If loaded successfully, start adding rasters
    if (e.Status == Esri.ArcGISRuntime.LoadStatus.Loaded)
    {
        AddRastersParameters parameters = new AddRastersParameters
        {
            InputDirectory = @"..\incoming\data\rasters"
        };
        rasterMosaic.AddRastersAsync(parameters);
    }
};

rasterMosaic.LoadAsync();
```

Copyright © 1995-2020 Esri. All rights reserved.

Use the `AddRastersParameters` class to set the input directory containing the raster files to be added and control other properties of the mosaic, such as minimum and maximum pixel sizes.

Create a raster from an image service

Raster and image data can be shared as an image service using ArcGIS Server. An image service provides access to raster data through a web service. A single raster dataset or a mosaic dataset that contains a collection of raster datasets can be served as one image service. The mosaic dataset can dynamically process and mosaic the images on the fly. An image service supports accessing both the mosaicked image and its catalog, as well as individual rasters in the catalog. For more information on image services, see Key concepts for image services in the ArcGIS help.

Use the `ImageServiceRaster` class to work with image services in your ArcGIS Runtime app. You can create an `ImageServiceRaster` with the URL of an image service. And if you want to display the raster data on a map, create a raster layer for it.

`ImageServiceRaster` is a subclass of `Raster`, so any operation that can be applied to the `Raster` class, such as raster functions, can also be applied to `ImageServiceRaster`. Besides the common properties inherited from the `Raster` class, `ImageServiceRaster` has additional properties and capabilities, such as the image service metadata and rendering rules. Once the image service raster has been loaded, the metadata of the image service can be retrieved through the service info property, which is represented by `ArcGISImageServiceInfo`. However, only a subset of the metadata, including attribution text and a list of information on predefined service rendering rules, is exposed in ArcGIS Runtime. For more information on the metadata of an image service, see Image Service in the ArcGIS REST API help.

```
// Create an image service raster from a URI
ImageServiceRaster serviceRaster = new ImageServiceRaster(new
Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/CharlotteLAS/ImageServer"));

// When the image service raster loads, the service info and rendering rule info will be
available
serviceRaster.Loaded += (s, e) =>
{
    // Get service info
    ArcGISImageServiceInfo serviceInfo = serviceRaster.ServiceInfo;

    // Get rendering rule info list
    IReadOnlyList<RenderingRuleInfo> renderingRuleInfos = serviceInfo.RenderingRuleInfos;

    // Call a function to load the raster with one of the rules
    // (Note: the rendering rule cannot be set on an ImageServiceRaster that is already loaded)
    AddRasterWithRenderingRule(renderingRuleInfos[3]);
};

// Load the image service raster
serviceRaster.LoadAsync();
```

Create a raster from a raster function

Raster functions can be applied to a raster to process that data. This processing is not permanently applied to the data; instead, it is applied on the fly as the rasters are accessed.

ArcGIS Runtime supportsa subset of raster functions that the ArcGIS REST API supports.

The following example uses a raster function (stored in a local JSON file).:

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create a new hillshade raster function from a JSON definition (stored in a text file).
var hillshadeFunction = new RasterFunction(pathToHillshadeFunctionJSON);

// Get the function's arguments.
RasterFunctionArguments rasterFunctionArguments = hillshadeFunction.Arguments;

// Assuming rasterNames has 1 entry, set the input raster.
IReadOnlyList<string> rasterNames = rasterFunctionArguments.GetRasterNames();
var inputRaster = new Raster(pathToInputTIF);
rasterFunctionArguments.SetRaster(rasterNames[0], inputRaster);

// Create a new raster based on the function.
Raster raster = new Raster(hillshadeFunction);

// Create a RasterLayer to display the Raster, add it to the map.
RasterLayer rasterLayer = new RasterLayer(raster);
MyMapView.Map = new Map(new Basemap(rasterLayer));
```

Raster functions supported by ArcGIS Runtime

ArcGIS Runtime supports a subset of raster functions supported by ArcGIS REST API. The raster functions supported by ArcGIS Runtime are provided in this section, along with the syntax for using them. The syntax is close to the ArcGIS REST syntax for the same functions but is not exactly the same.

The general syntax for ArcGIS Runtime raster functions is the following:

```
{
  "raster_function":{"type":"<Raster Function Name>"},
  "raster_function_arguments":
  {
    "argument1":"<JSON Value Object>",
    "argument2":"<JSON Value Object>",
    "argumentN":"<JSON Value Object>",
    "type":"Raster_function_arguments"
  },
  "type":"Raster_function_template"
}
```

Clip

```
{
  "raster_function_arguments":
  {
    "minx":{"double":value,"type":"Raster_function_variable"},
    "miny":{"double":value,"type":"Raster_function_variable"},
    "maxx":{"double":value,"type":"Raster_function_variable"},
    "maxy":{"double":value,"type":"Raster_function_variable"},
    "dx":{"double":cell_size_x,"type":"Raster_function_variable"},
    "dy":{"double":cell_size_y,"type":"Raster_function_variable"},
    "raster":{"name":"raster","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "raster_function":{"type":"Clip_function"},
  "type":"Raster_function_template"
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Colormap

```
{
  "raster_function_arguments":
  {
    "raster_colormap":{"colors":[color1,color2,...,colorN],"type":"Raster_function_variable"},
    "raster":{"name":"raster","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "raster_function":{"type":"Colormap_function"},
  "type":"Raster_function_template"
}
```

Colormap_to_RGB

```
{
  "raster_function":{"type":"Colormap_to_RGB_function"},
  "raster_function_arguments":
  {
    "raster":{"name":"raster","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "type":"Raster_function_template"
}
```

Color_ramp

```
{
  "raster_function":{"type":"Color_ramp_function"},
  "raster_function_arguments":
  {
    "resizable":{"bool":false,"type":"Raster_function_variable"},
    "color_ramp":
    {
      "color_ramp":
      {
        "ramps":
        [

{"to_color":[0,255,0],"from_color":[0,191,191],"num_colors":3932,"type":"Algorithmic_color_ramp","algorith

{"to_color":[255,255,0],"from_color":[0,255,0],"num_colors":3932,"type":"Algorithmic_color_ramp","algorith

{"to_color":[255,127,0],"from_color":[255,255,0],"num_colors":3932,"type":"Algorithmic_color_ramp","algori

{"to_color":[191,127,63],"from_color":[255,127,0],"num_colors":3932,"type":"Algorithmic_color_ramp","algor

{"to_color":[20,20,20],"from_color":[191,127,63],"num_colors":3935,"type":"Algorithmic_color_ramp","algori
        ],
        "type":"Multipart_color_ramp"
      },
      "type":"Raster_function_variable"
    },
    "raster":{"name":"raster","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "type":"Raster_function_template"
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Composite_band

```
{
  "raster_function":{"type":"Composite_band_function"},
  "raster_function_arguments":
  {

"raster_names":{"name":"raster_names","string_array":["r1","r2","r3","r4"],"type":"Raster_function_variabl
    "r1":{"name":"r1","is_raster":true,"type":"Raster_function_variable"},
    "r2":{"name":"r2","is_raster":true,"type":"Raster_function_variable"},
    "r3":{"name":"r3","is_raster":true,"type":"Raster_function_variable"},
    "r4":{"name":"r4","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "type":"Raster_function_template"
}
```

Extract_band

```
{
  "raster_function_arguments":
  {
    "exact_match":{"bool":false,"type":"Raster_function_variable"},
    "band_indexes":{"int_array":[2,1,0],"type":"Raster_function_variable"},
    "raster":{"name":"raster","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "raster_function":{"type":"Extract_band_function"},
  "type":"Raster_function_template"
}
```

Geometric

```
{
  "raster_function":{"type":"Geometric_function"},
  "raster_function_arguments":
  {
    "raster_transform":{"raster_transform":"Raster Transform JSON
object","type":"Raster_function_variable"},
    "z_offset":{"double":0,"type":"Raster_function_variable"},
    "z_factor":{"double":1,"type":"Raster_function_variable"},
    "raster":{"is_raster":true,"name":"raster","type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "type":"Raster_function_template"
}
```

Hillshade

```
{
  "raster_function":{"type":"Hillshade_function"},
  "raster_function_arguments":
  {
    "z_factor":{"double":0.0002,"type":"Raster_function_variable"},
    "slope_type":{"raster_slope_type":"none","type":"Raster_function_variable"},
    "azimuth":{"double":315,"type":"Raster_function_variable"},
    "altitude":{"double":45,"type":"Raster_function_variable"},
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    "nbits":{"int":8,"type":"Raster_function_variable"}, // Number of bits per pixel for output
raster
    "raster":{"name":"raster","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "type":"Raster_function_template"
}
```

Mask

```
{
  "raster_function":{"type":"Mask_function"},
  "raster_function_arguments":
  {
    "nodata_values":{"double_array":[value1, value2, ...,
valueN],"type":"Raster_function_variable"},
    "nodata_interpretation":{"nodata_interpretation":"all","type":"Raster_function_variable"},
    "raster":{"name":"raster","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "type":"Raster_function_template"
}
```

Pansharpen

```
{
  "raster_function":{"type":"Pansharpen_function"},
  "raster_function_arguments":
  {

"weights":{"double_array":[0.10000000000000001,0.5,0.40000000000000002,0.29999999999999999],"type":"Raster
    "pansharpen_type":{"pansharpen_type":"gram_schmidt","type":"Raster_function_variable"},
    "pan_raster":{"name":"pan_raster","is_raster":true,"type":"Raster_function_variable"},
    "raster":{"name":"raster","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "type":"Raster_function_template"
}
```

Raster_calculator

```
{
  "raster_function_arguments":
  {
    "expression":{"string":"r1 + r2","name":"expression","type":"Raster_function_variable"},
    "raster_names":{"name":"raster_names","type":"Raster_function_variable"},

"raster_names":{"name":"raster_names","string_array":["r1","r2"],"type":"Raster_function_variable"},
    "r1":{"name":"r1","is_raster":true,"type":"Raster_function_variable"},
    "r2":{"name":"r2","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "raster_function":{"type":"Raster_calculator_function"},
  "type":"Raster_function_template"
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Stretch

```
{
  "raster_function":{"type":"Stretch_function"},
  "raster_function_arguments":
  {
    "stretch_type":{"raster_stretch_type":"minimum_maximum","type":"Raster_function_variable"},
    "min_values":{"double_array":[-10977],"type":"Raster_function_variable"},
    "max_values":{"double_array":[8685],"type":"Raster_function_variable"},
    "estimate_stats":{"bool":false,"type":"Raster_function_variable"},
    "raster":{"name":"raster","is_raster":true,"type":"Raster_function_variable"},
    "type":"Raster_function_arguments"
  },
  "type":"Raster_function_template"
}
```

Add a raster using a raster layer

It's not necessary to display the raster data you work with in your app. However, if you want your users to view raster data on a map, add the raster using the `RasterLayer` class. `RasterLayer` can render raster data from any type of `Raster`.

You can add it to a map as either a basemap or an operational layer. When adding a raster layer as an operational layer to a map with different spatial reference, the layer will be reprojected on the fly and be added to the map.

You can also change the renderer, as described below, to control how the data is visualized on the map.

See Layers and tables for more information about creating and working with raster layers.

Supported raster formats

ArcGIS Runtime supports a subset of raster file formats that ArcGIS Desktop supports. The raster file formats ArcGIS Runtime supports include the following:

- ASRP/USRP
- CRF
- DTED0, 1, 2
- GeoTIFF
- HFA
- HRE
- IMG
- JPEG
- JPEG 2000
- MrSID, generations 2, 3, and 4
- NITF
- PNG
- RPF (CIB)
- RPF (CADRG)
- SRTM1, 2

Copyright © 1995-2020 Esri. All rights reserved.

- Mobile mosaic datasets

Copyright © 1995-2020 Esri. All rights reserved.

Display KML content

Keyhole Markup Language (KML) is a geographic data format popularized by Google Earth. KML files can be distributed with supporting content, including images and 3D models, in a KMZ archive. ArcGIS Runtime supports version 2.2 of the KML specification as defined by the Open GIS Consortium (OGC).

Display a KML/KMZ file

KML content is loaded using a `KmlDataset`. The `KmlDataset` constructor takes a URL, which can point to a local file or a network location. Loading the layer will cause the associated dataset to load. KML layers can also be loaded from portal items, which can specify a link to a file or the KML/KMZ file itself.

```
KmlDataset kmlDataSource = new KmlDataset(kmlFilePathUri);
KmlLayer kmlOperationalLayer = new KmlLayer(kmlDataSource);
```

Note:
See Licensing your ArcGIS Runtime App to learn more about the licensing requirements for KML files loaded over the network and from disk.

Note:
Creating, editing, and saving KML data in ArcGIS Runtime is described in more detail in the topic Edit KML content.

Many KML files are wrappers that point to resources over the network. For example, a weather map might consist of a single network link that points to the latest forecast, to be retrieved every 5 minutes. As a result, loading the layer does not necessarily guarantee that the content has loaded – linked content can fail to load without affecting the load status of the layer.

Note:
Many KML files, even those delivered over secure HTTPS connections, point to resources via insecure HTTP links. These resources may fail to load as a consequence of App Transport Security on iOS. Add appropriate ATS exceptions as needed for KML content. See Apple's developer documentation for more information about ATS exceptions.

Explore the KML content tree

KML layers contain content in a hierarchy. You may need to programmatically explore this hierarchy to interact with KML content. For example, to turn off a screen overlay, you would need to first find it in the tree, then change its visibility. This code will follow a recursive pattern, using a function that calls itself for each node in the tree. The KML tree should be explored starting with `KmlDataset`. `KmlDataset` exposes the KML feature tree with the `RootNodes` property. This provides a collection of `KmlNodes`. Start with a function that accepts an enumeration of `KmlNodes`.

```
private void ExploreKml(IEnumerable<KmlNode> nodes)
{
    foreach(KmlNode node in nodes)
    {
        // Do work here...
    }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

The non-recursive part of turning off a screen overlay is toggling its visibility once it's found. Because you're working with a generic `KmlNode`, you first need to determine whether each iterated node is a screen overlay, `KmlScreenOverlay`. After the screen overlay is found, toggle its visibility.

```
private void ExploreKml(IEnumerable<KmlNode> nodes)
{
    foreach(KmlNode node in nodes)
    {
        if (node.GetType() == typeof(KmlScreenOverlay))
        {
            node.IsVisible = !node.IsVisible;
        }
    }
}
```

You should then look for screen overlay nodes that may exist deeper in the hierarchy, however. Recursively call `exploreKml` on any nodes that have children.

```
private void ExploreKml(IEnumerable<KmlNode> nodes)
{
    // Iterate all nodes.
    foreach (KmlNode node in nodes)
    {
        // Toggle node visibility for any screen overlays found.
        if (node.GetType() == typeof(KmlScreenOverlay))
        {
            node.IsVisible = !node.IsVisible;
        }

        // See if this node has child nodes; store them in a list.
        List<KmlNode> children = new List<KmlNode>();
        if (node is KmlContainer containerNode)
        {
            children.AddRange(containerNode.ChildNodes);
        }
        if (node is KmlNetworkLink networkLinkNode)
        {
            children.AddRange(networkLinkNode.ChildNodes);
        }

        // Recursively call this function with the child node list.
        ExploreKml(children);
    }
}
```

You can call the recursive function initially with`KmlDataset`'s `RootNodes`, either after loading the map or in reaction to user action (like a button press).

The tree-exploration pattern is useful for other tasks involving KML. For example, ArcGIS Earth uses a recursive pattern to build a table of contents:

Copyright © 1995-2020 Esri. All rights reserved.

Identify and popups in KML

In the ArcGIS information model, a popup is defined with a set of fields that describe an object, including how that information is formatted. Unlike ArcGIS feature services, KML files don't define a standard schema for object attributes. Instead, KML files may provide a rich HTML annotation for each object, which can be presented in place of a popup. You can access the the HTML annotation via `KmlNode.BalloonContent`, then display that using a webview.

Viewpoints in KML

KML has two primary ways for defining viewpoints:

- LookAt - defines a camera relative to the position of a KML feature
- Camera - defines the position of the camera explicitly

`KmlViewpoint` can represent both types of viewpoints. Custom code is required to convert from a KML viewpoint to an ArcGIS Runtime viewpoint, which can be used for navigating a scene. See Google's reference documentation for details, including diagrams. Earth browsing apps should respect LookAt viewpoints specified in KML content.

```
private async void NavigateToNode(KmlNode node)
{
    // Get a corrected Runtime viewpoint using the KmlViewpoint.
    Viewpoint runtimeViewpoint = ViewpointFromKmlViewpoint(node, out bool
viewpointNeedsAltitudeAdjustment);
    if (viewpointNeedsAltitudeAdjustment)
    {
        runtimeViewpoint = await GetAltitudeAdjustedViewpointAsync(node, runtimeViewpoint);
    }

    // Set the viewpoint.
    if (runtimeViewpoint != null && !runtimeViewpoint.TargetGeometry.IsEmpty)
    {
        await MySceneView.SetViewpointAsync(runtimeViewpoint);
    }
}

private Viewpoint ViewpointFromKmlViewpoint(KmlNode node, out bool needsAltitudeFix)
{
    KmlViewpoint kvp = node.Viewpoint;
    // If KmlViewpoint is specified, use it.
    if (kvp != null)
    {
        // Altitude adjustment is needed for everything except Absolute altitude mode.
        needsAltitudeFix = (kvp.AltitudeMode != KmlAltitudeMode.Absolute);
        switch (kvp.Type)
        {
            case KmlViewpointType.LookAt:
                return new Viewpoint(kvp.Location,
                    new Camera(kvp.Location, kvp.Range, kvp.Heading, kvp.Pitch, kvp.Roll));
            case KmlViewpointType.Camera:
                return new Viewpoint(kvp.Location,
                    new Camera(kvp.Location, kvp.Heading, kvp.Pitch, kvp.Roll));
```

Copyright © 1995-2020 Esri. All rights reserved.

```
            default:
                throw new InvalidOperationException("Unexpected KmlViewPointType: " + kvp.Type);
        }
    }

    if (node.Extent != null && !node.Extent.IsEmpty)
    {
        // When no altitude specified, assume elevation should be taken into account.
        needsAltitudeFix = true;

        // Workaround: it's possible for "IsEmpty" to be true but for width/height to still be
zero.
        if (node.Extent.Width == 0 && node.Extent.Height == 0)
        {
            // Defaults based on Google Earth.
            return new Viewpoint(node.Extent, new Camera(node.Extent.GetCenter(), 1000, 0, 45,
0));
        }
        else
        {
            Envelope tx = node.Extent;
            // Add padding on each side.
            double bufferDistance = Math.Max(node.Extent.Width, node.Extent.Height) / 20;
            Envelope bufferedExtent = new Envelope(
                tx.XMin - bufferDistance, tx.YMin - bufferDistance,
                tx.XMax + bufferDistance, tx.YMax + bufferDistance,
                tx.ZMin - bufferDistance, tx.ZMax + bufferDistance,
                SpatialReferences.Wgs84);
            return new Viewpoint(bufferedExtent);
        }
    }
    else
    {
        // Can't fly to.
        needsAltitudeFix = false;
        return null;
    }
}

// Asynchronously adjust the given viewpoint, taking into consideration elevation and KML
altitude mode.
private async Task<Viewpoint> GetAltitudeAdjustedViewpointAsync(KmlNode node, Viewpoint
baseViewpoint)
{
    // Get the altitude mode; assume clamp-to-ground if not specified.
    KmlAltitudeMode altMode = KmlAltitudeMode.ClampToGround;
    if (node.Viewpoint != null)
    {
        altMode = node.Viewpoint.AltitudeMode;
    }

    // If the altitude mode is Absolute, the base viewpoint doesn't need adjustment.
    if (altMode == KmlAltitudeMode.Absolute)
    {
        return baseViewpoint;
    }

    double altitude;
    Envelope lookAtExtent = baseViewpoint.TargetGeometry as Envelope;
    MapPoint lookAtPoint = baseViewpoint.TargetGeometry as MapPoint;

    if (lookAtExtent != null)
    {
```

Copyright © 1995-2020 Esri. All rights reserved.

```
        // Get the altitude for the extent.
        try
        {
            altitude = await
MySceneView.Scene.BaseSurface.GetElevationAsync(lookAtExtent.GetCenter());
        }
        catch (Exception)
        {
            altitude = 0;
        }

        // Apply elevation adjustment to the geometry.
        Envelope target;
        if (altMode == KmlAltitudeMode.ClampToGround)
        {
            target = new Envelope(
                lookAtExtent.XMin, lookAtExtent.YMin,
                lookAtExtent.XMax, lookAtExtent.YMax,
                altitude, lookAtExtent.Depth + altitude,
                lookAtExtent.SpatialReference);
        }
        else
        {
            target = new Envelope(
                lookAtExtent.XMin, lookAtExtent.YMin,
                lookAtExtent.XMax, lookAtExtent.YMax,
                lookAtExtent.ZMin + altitude, lookAtExtent.ZMax + altitude,
                lookAtExtent.SpatialReference);
        }

        if (node.Viewpoint != null)
        {
            // Return adjusted geometry with adjusted camera if a viewpoint was specified on the
node.
            return new Viewpoint(target, baseViewpoint.Camera.Elevate(altitude));
        }
        else
        {
            // Return adjusted geometry.
            return new Viewpoint(target);
        }
    }
    else if (lookAtPoint != null)
    {
        // Get the altitude adjustment.
        try
        {
            altitude = await MySceneView.Scene.BaseSurface.GetElevationAsync(lookAtPoint);
        }
        catch (Exception)
        {
            altitude = 0;
        }

        // Apply elevation adjustment to the geometry.
        MapPoint target;
        if (altMode == KmlAltitudeMode.ClampToGround)
        {
            target = new MapPoint(lookAtPoint.X, lookAtPoint.Y, altitude,
lookAtPoint.SpatialReference);
        }
        else
        {
```

Copyright © 1995-2020 Esri. All rights reserved.

```
            target = new MapPoint(
                lookAtPoint.X, lookAtPoint.Y, lookAtPoint.Z + altitude,
                lookAtPoint.SpatialReference);
        }

        if (node.Viewpoint != null)
        {
            // Return adjusted geometry with adjusted camera if a viewpoint was specified on the
node.
            return new Viewpoint(target, baseViewpoint.Camera.Elevate(altitude));
        }
        else
        {
            // Google Earth defaults: 1000m away and 45-degree tilt.
            return new Viewpoint(target, new Camera(target, 1000, 0, 45, 0));
        }
    }
    else
    {
        throw new InvalidOperationException("KmlNode has unexpected Geometry for its Extent: " +
                                            baseViewpoint.TargetGeometry);
    }
}
```

Play KML tours

ArcGIS Runtime supports playing tours contained in KML files. A KML tour defines a map exploration experience with narration and music, animated movement between viewpoints, and dynamic animation of KML feature properties.

To play a KML tour, create a KML tour controller and set its `Tour` property to the tour you want to play. Then use the `Play`, `Pause`, and `Restart` methods on `KmlTourController` to work with the tour. If the tour contains audio, that audio will play automatically with the tour. To monitor the status of the tour, refer to the `KmlTour.TourStatus` property.

The following example shows a function that can search KML nodes for a tour and assign the first one it finds to a `KmlTourController`.

```
// The KML tour controller provides player controls for KML tours.
private readonly KmlTourController _tourController = new KmlTourController();

// A function to search KML nodes and find a tour.
private void FindKmlTour(IEnumerable<KmlNode> rootNodes)
{
    // Hold a list of nodes to explore.
    Queue<KmlNode> nodesToExplore = new Queue<KmlNode>();

    // Add each root node to the list.
    foreach (KmlNode rootNode in rootNodes)
    {
        nodesToExplore.Enqueue(rootNode);
    }

    // Keep exploring until a tour is found or there are no more nodes.
    while (nodesToExplore.Any())
    {
        // Remove a node from the queue.
        KmlNode currentNode = nodesToExplore.Dequeue();

        // If the node is a tour, assign it to the tour controller.
        if (currentNode is KmlTour)
```

Copyright © 1995-2020 Esri. All rights reserved.

```
        {
            _tourController.Tour = (KmlTour) currentNode;
            return;
        }

        // If the node is a container, add all of its children to the list of nodes to explore.
        if (currentNode is KmlContainer)
        {
            KmlContainer container = (KmlContainer) currentNode;
            foreach (KmlNode node in container.ChildNodes)
            {
                nodesToExplore.Enqueue(node);
            }
        }
    }
}
```

Learn more

- Google: Touring in KML
- Sample: Play a KML tour
- Guide: Edit KML content

Copyright © 1995-2020 Esri. All rights reserved.

Display a grid

A grid is a collection of horizontal and vertical lines which can be rendered over the top of your map view to help show the location of the current view point.

You can add the following types of grid to your map view:

- Latitude and longitude grid - also known as a graticule
- USNG - United States National Grid
- MGRS - Military Grid Reference System
- UTM - a grid showing top level UTM zones

Add a grid to your map view

To add a grid to your map view, first create an instance of the type of grid you want to display (such as a UTM grid) and set the grid property on the map view to the new grid instance.

```
// create a grid for showing Latitude and Longitude (Meridians and Parallels)
LatitudeLongitudeGrid grid = new LatitudeLongitudeGrid();

// display the grid on the map view
MyMapView.Grid = grid;
```

Controlling the label and grid visibility

The grid visibility can be controlled by using the grid's visible property.

```
MyMapView.Grid.IsVisible = showGrid;
```

In addition, you can control the visibility of the labels on your grid using the label visible property.

Copyright © 1995-2020 Esri. All rights reserved.

```
MyMapView.Grid.IsLabelVisible = showLabels;
```

Change the position of the grid labels

You can change the default positioning of the labels on your grid by changing the label position property to one of the following values:

- Geographic
- Bottom left
- Bottom right
- Top left
- Top right
- Center
- All sides

For example, you could specify that the grid labeling be placed geographically. This will result in labels anchored to a geographical position on the map view.

```
MyMapView.Grid.LabelPosition = GridLabelPosition.Geographic;
```

Alternatively, you may choose to position the labels so they are always on the top and right sides of your map view.

```
MyMapView.Grid.LabelPosition = GridLabelPosition.TopRight;
```

Copyright © 1995-2020 Esri. All rights reserved.

If you are positioning your labels against the top, bottom, left or right sides of your map views, you can set the offset distance between the label and the edge of the map by setting the label offset property.

Label positioning works in ways appropriate for different types of grids.

- The latitude longitude grid respects the label position property.
- MGRS and USNG grids define zonal resolution levels: the first level uses UTM zones; the second uses 100 km squares, each positioned in the center of the zone. As zooming continues, the UTM and 100 km identifiers are concatenated and positioned according to the label position property.
- The UTM grid has a single level showing grid zones using identifiers like **30U** or **11S**. These labels are not affected by the label position property.

Changing the style of the grid lines and labels

The default styling of the grid line and labels is designed to be easily visible against most of the standard base maps, however you can optionally set your own styling against any given level of resolution of the grid. You can set the same styling for all levels of grid resolution, or use a different styling for various levels.

Before you can apply a new line or label style, you need to find the number of grid resolution levels for the type of grid you are displaying. This is found by querying the level count property.

```
// find the number of resolution levels in the grid
var gridLevels = MyMapView.Grid.LevelCount;
```

The example below shows how to style your grid so it has purple lines and white labels with a blue halo.

Copyright © 1995-2020 Esri. All rights reserved.

Styling grid lines and grid labels is done by applying line symbols and text symbols, respectively. The characteristics of the symbols are used to style the lines and label. First, you first need to create line and text symbols to apply to your grid.

```
// create a line symbol (a thick purple line)
SimpleLineSymbol lineSymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Solid, Colors.Purple,
5);

// create a text style with white text and blue halo.
TextSymbol textSymbol = new TextSymbol();
textSymbol.Size = 20;
textSymbol.Color = Colors.DarkBlue;
textSymbol.HaloColor = Colors.White;
textSymbol.HaloWidth = 5;
```

Then you can loop through each of the grid levels and apply the symbols to style your grid.

```
// apply the line and text symbols to each level in the grid
for (var level = 0;level< gridLevels; level++)
{
    grid.SetTextSymbol(level, textSymbol);
    grid.SetLineSymbol(level, lineSymbol);
}
```

Some of the grid types have additional properties to allow you to set the units of measure or the format of the labels. For example, the latitude longitude grid has a label style property which allows you to choose whether to label your grids using decimal degrees or formatted degrees, minutes and seconds.

Copyright © 1995-2020 Esri. All rights reserved.

```
// format the labels to display in degrees, minutes and seconds
grid.LabelFormat = LatitudeLongitudeGridLabelFormat.DegreesMinutesSeconds;
```

Tip:

To work with any of these types of coordinates separately from a grid, you can use `CoordinateFormatter` to convert between a `MapPoint` and a string formatted to contain any of these coordinate notations.

Supported spatial references for grid

The grids are supported to work in spatial references which are continuously pannable or some polar spatial references.

You can check the is pannable property on your spatial reference class to see if is pannable. Examples of continuously pannable spatial references include WGS84 (WKID 4326) and Web Mercator Auxiliary Sphere (WKID 102113, 102100, or 3857).

The following polar spatial references are also supported:

- North pole Azimuthal Equidistant (WKID 102016)
- North pole gnomic (WKID 102034)
- North pole lambert equal area (WKID 102017)
- North pole orthographic (WKID 102035)
- North pole stereographic (WKID 102018)
- NSIDE EASE North (WKID 3408)
- NSIDC Sea Ice polar stereographic north (WKID 3411)
- WGS 1984 NSIDC Sea Ice Polar Stereographic North (WKID 3413)
- UPS North (WKID 32661)
- South pole Azimuthal Equidistant (WKID 102019)
- South pole gnomic (WKID 102036)
- South pole lambert equal area (WKID 102020)
- South pole orthographic (WKID 102037)
- South pole stereographic (WKID 102021)
- NSIDE EASE South (WKID 3409)
- NSIDC Sea Ice polar stereographic South (WKID 3412)
- UPS South (WKID 32761)
- WGS1984 Antartic polar stereographic (WKID 3031)

Copyright © 1995-2020 Esri. All rights reserved.

Show the device's location

Most devices provide information about their location. Each platform has API interfaces on the device that abstract how the location is retrieved, whether through Wi-Fi, cellular networks, or GPS.

Using the map view's location display and a location data source, the device's location can be displayed on top of a map. You can respond to changes in this location to find nearby points of interest, provide navigation instructions, and so on.

Display the location on a map

You can display the current location of the device in a map view by using its location display. The map view location display handles getting the current location from the location data source and displaying it on top of the map.

Additional settings for the location display include:

- Automatic panning to keep the latest point location visible
- Change of symbol based on the status of the device's location
- Display of metadata, for example the estimated accuracy of the location
- Set the zoom scale of the map in automatic panning or navigation modes
- Set opacity for the symbols used to show location and accuracy

Current location display

By default, the location display uses a blue, round symbol for the current location. Depending on signal strength, satellite positions, and other factors, the accuracy of the reported location can vary. An additional blue circle around the location symbol indicates the estimated range of accuracy for the current location. As locations are read from the configured data source, this outer circle may contract and expand as accuracy increases or decreases.

As the device moves, the map responds to location updates and appropriately updates the location symbol's position.

Location data sources

Location display uses a location data source to provide the current position of the device. By default, location display uses the system's location provider. You can create custom location data sources to process input from other location providers, such as an external GPS receiver. To simulate location updates with a collection of defined points for testing purposes, use a `SimulatedLocationDataSource`.

Simulate device location updates

You can use a simulated location data source to test or demonstrate functionality that requires a set of locations without having to rely on incoming GPS information. This is especially useful for testing functionality that listens for location changes, such as navigating a route. With a simulated location data source, you can define a collection of mock locations to visit in a few ways:

- Vertices along a polyline—The vertices of a provided polyline define locations and the velocity of each is calculated based on its distance from the previous one. The current time is set as the start time for the first location and each subsequent location's time is incremented by one second.
- Points generated along a polyline—Generates locations using a provided polyline and a velocity. The specified start time is used for the first location and each subsequent location's time is incremented by one second.
- A defined set of points—Locations are defined explicitly.

Copyright © 1995-2020 Esri. All rights reserved.

After the points are defined, enable the simulated location data source to start the iteration of points in the collection. A location change event is raised for each point. Locations are visited in the order in which they appear in the collection, allowing you to simulate travel along a path at a defined rate of speed. You can use the index of the location in the array to get the current location or to move to a specific location in the array. This allows you to skip over or to return to previous locations.

The default rate at which the locations are iterated in the simulated location data source is one per second. You can change the frequency of location updates by changing the iteration rate. A rate of 2.0, for example, doubles the frequency of location updates and a rate of 0.5 cuts the frequency of updates by half. Note that this does not change the time stamps on the locations, which are always one second apart.

The following example shows a simulated data source to follow a path defined by a polyline. Each point along the line at 30 meter intervals is visited.

```
// Create a simulated location data source
SimulatedLocationDataSource simulatedLocationData = new SimulatedLocationDataSource();

// Create simulation parameters to specify a rate of travel of 30 meters/second.
SimulationParameters simulatedLocationParams = new SimulationParameters
{
    Velocity = 30.0 // meters per second
};

// Set the simulated locations using a polyline (a route's geometry) and the simulation
parameters.
simulatedLocationData.SetLocationsWithPolyline(route.RouteGeometry, simulatedLocationParams);

// Assign the simulated location data source to the map view's location display and enable it.
// Location updates are handled in MyMapView.LocationDisplay.LocationChanged
MyMapView.LocationDisplay.DataSource = simulatedLocationData;
MyMapView.LocationDisplay.IsEnabled = true;
```

Autopan modes

The map provides a number of presets that define how the map behaves when location updates are received:

- Off
- Recenter
- Navigation
- Compass navigation

Off mode

In this mode, the map only updates the position of the location symbol on the map. It does not recenter. Therefore, it is possible that the location symbol may move off the screen in response to location updates.

Recenter mode

In this mode, the map attempts to keep the location symbol on-screen by recentering the location symbol when the symbol moves outside a wander extent. Therefore, the location symbol may move freely within the wander extent, but as soon as the symbol exits the wander extent, the map view pans to recenter itself on the symbol.

By default, the wander extent is half the size of the map's viewable extent, but it can be customized. A value of 1 implies a wander extent equal to the size of the map's viewable extent, and the location symbol may move up to the edge of the map before the map recenters. A value of 0 implies an infinitesimal wander extent, and the map recenters on every location update.

Copyright © 1995-2020 Esri. All rights reserved.

If a user pans the map in this mode, the mode automatically changes its Autopan mode to Off, preventing the map from unwanted recentering based on future location updates.

Navigation mode

This mode is best for in-vehicle navigation. In this mode, the location symbol is pinned to a point on the screen and always points toward the top edge of the device. The map pans and rotates based on location updates and the direction the device is traveling in. Therefore, the location symbol appears stationary, while the map underneath it appears to be moving and rotating.

The position of the location symbol can be adjusted. A value of 0 implies the location symbol should be positioned at the bottom edge of the map, and 1 implies the top edge. A value between 0 and 1 positions the symbol somewhere along the centerline from the bottom edge to the top edge.

If a user pans the map in this mode, the mode automatically changes to Off.

Compass Navigation mode

This mode is better when the user is walking. As in Navigation mode, the location symbol is pinned to a point on the screen and always points toward the top edge of the device. However, the map rotates based on the user's bearing relative to magnetic north, and not based on the direction the device travels in, which is the case in Navigation mode. Therefore, the map reflects what lies ahead of the user as the user pivots around their location.

The location symbol position can be adjusted. A value of 0 implies the location symbol should be positioned at the bottom edge of the map, and 1 implies the top edge. A value between 0 and 1 positions the symbol somewhere between the bottom and top edges.

If a user pans the map in this mode, the mode automatically changes to Off.

Customizing the location symbol

The SDK provides default images that are used for the location symbol. These images are resources embedded in the ArcGIS Runtime assembly.

The following properties on `LocationDisplay` can be changed to use your own symbols.

- `AccuracySymbol` — fill symbol that illustrates the accuracy of the current location
- `AcquiringSymbol` — while the location is being acquired
- `HeadingSymbol` — while in compass navigation mode
- `CourseSymbol` — when the location is moving; the symbol rotates to show the current heading
- `DefaultSymbol` — while the location is not moving

Depending on the autopan mode in use, and the type of information provided by the location updates, one of the default images will be used for the location symbol.

In the off, recenter, and navigation modes, the default image is used only when location information is available. The course symbol is used if both location information and course (direction of travel) information are available. In the off and recenter modes, the course symbol is appropriately rotated by an amount specified by the course to display the direction in which the device is traveling; however, in Navigation mode, the map rotates instead.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
Course information is only available when the device is moving at some nontrivial velocity. This is why the map typically only displays the course symbol when the device is traveling in a vehicle.

In the Compass Navigation mode, the heading image is used to display the device's location and bearing.

Copyright © 1995-2020 Esri. All rights reserved.

Visualize and compare data over time

Several layers, including `FeatureLayer` and `ArcGISMapImageLayer` are time-aware, meaning they can use time-based rendering and filtering if it's enabled on the data source at the time of publishing. The SDK provides several ways to work with time:

- Filter data with a time extent applied to the geo view
- Apply a time offset to the data in a layer (useful for comparing data over time)
- Control which eligible layers participate in time filtering
- Add time-based parameters to queries

Filter data by time

You can filter data in time-aware layers by defining a time extent and applying it to the map or scene view. All time-aware layers in the map (or scene) displayed by the view will filter their data per the view's time extent (except when time filtering is not supported or is disabled).

To filter data by time, first create a time extent that captures the desired time range. For example, to include only data for the last 60 days, create two `DateTimeOffset` `Date`s, one for now and one for now minus 60 days:

```
DateTimeOffset now = DateTime.Now;
DateTimeOffset sixtyDaysAgo = DateTime.Now - new TimeSpan(60,0,0,0);
TimeExtent extent = new TimeExtent(sixtyDaysAgo, now);
```

Note:
The start time must be less than the end time - otherwise, an exception will be thrown.

Next, apply the time extent to the view.

```
MyMapView.TimeExtent = extent;
```

After applying the time extent to the view, all applicable layers in the map or scene contained by the view will only show data from the specified time extent. This includes layers that:

- Support time filtering
- Are connected to time-enabled data sources
- Have not had time filtering disabled

You can explicitly exclude layers from participating in time filtering.

```
myLayer.IsTimeFilteringEnabled = false;
```

You can also detect if an otherwise time-aware layer is incapable of working with time (e.g. because it is connected to a data source without time values).

```
if (myLayer.SupportsTimeFiltering)
{
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    ... // Time-dependent work
}
```

Comparing two time periods

Because the time extent is defined for a geo view (map view or scene view), it is applied uniformly to all of the view's layers. Using a time offset, you can add (or subtract) a given amount of time from the filter defined by the geo view. The `TimeOffset` property of time-aware layers is useful for offsetting the filter applied to participating layers. For example, you can use a one-year offset to visualize year-to-year changes in a layer's data.

To visually compare data over time, first create two layers from the same service.

```
Uri mapServerUri = new Uri("...");
ArcGISMapImageLayer baseImageryLayer = new ArcGISMapImageLayer(mapServerUri);
ArcGISMapImageLayer offsetImageryLayer = new ArcGISMapImageLayer(mapServerUri);
```

Next, apply a time offset to one layer.

```
offsetImageryLayer.TimeOffset = new TimeValue(1, TimeUnit.Years);
```

Add the layers to a map and the map to a map view.

```
Map myMap = new Map(Basemap.CreateTopographic());
myMap.OperationalLayers.Add(baseImageryLayer);
myMap.OperationalLayers.Add(offsetImageryLayer);
MyMapView.Map = myMap;
```

Set a time extent on the map view to filter the data.

```
MyMapView.TimeExtent = new TimeExtent(start, end);
```

Compare layers with and without time filtering

You can set the `IsTimeFilteringEnabled` property on a layer to prevent it from filtering its content based on the time extent defined on the geo view. To show both filtered and unfiltered data, first add two layers to the map.

```
Uri firstUri = new Uri("...");
Uri otherUri = new Uri("...");
FeatureLayer timeFilteredLayer = new FeatureLayer(firstUri);
FeatureLayer unfilteredLayer = new FeatureLayer(otherUri);
```

Next, exclude the unfiltered layer from time filtering.

```
unfilteredLayer.IsTimeFilteringEnabled = false;
```

Add the layers to a map and the map to a map view.

```
Map myMap = new Map(Basemap.CreateTopographic());
myMap.OperationalLayers.Add(filteredLayer);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
myMap.OperationalLayers.Add(unfilteredLayer);
MyMapView.Map = myMap;
```

Set the time extent on the map view to filter the data.

```
MyMapView.TimeExtent = new TimeExtent(startTime, endTime);
```

Copyright © 1995-2020 Esri. All rights reserved.

Display electronic navigational charts

Electronic navigational charts (ENCs) are georeferenced vector datasets for the visualization and analysis of hydrographic and maritime information. ArcGIS Runtime supports ENCs that conform to the International Hydrographic Organization (IHO) S-57 standard. Use ENCs in your ArcGIS Runtime apps to:

- Visualize S-57 data in compliance with S-52 standards and specifications for chart content display
- Use S-57 data as an additional data source for situational awareness and decision making
- Combine S-57 datasets with other sources of information for geospatial analysis

Get started

ENC functionality in runtime is accessed via the `Esri.ArcGISRuntime.Hydrography` NuGet package. The NuGet package will ensure that all binaries and resources required by runtime for ENC functionality are deployed to your application's output folder.

Understand electronic navigational charts (ENCs)

An ENC cell represents a navigational chart of a rectangular geographic area at a particular scale. Cells are bounded by meridians and parallels, though the actual area of coverage contained in the cell may be any shape. All the data in a single cell corresponds to a single navigational use, such as overview, general, or coastal. (A navigational use corresponds to a scale range suitable for a particular use.)

Cells are stored in a single base dataset file and zero or more update dataset files. Each dataset file has a unique name. A base dataset file plus its update files (if any) when loaded together comprise the updated geographic data of one cell.

ENC data is distributed in exchange sets, which can contain many cells. On the file system, each exchange set resides in its own folder named `ENC_ROOT`. Each `ENC_ROOT` folder contains a collection of dataset files (`*.000`), update files (`*.001-*.999`), a catalog file containing metadata about the exchange set (`CATALOG.031`), and other optional files with data referenced by the exchange set, such as text and image files.

See S-57 Appendix B for additional details about S-57 ENC exchange sets and their content.

Display an electronic navigational chart

Access an exchange set on the file system using an `EncExchangeSet` object. An exchange set can be loaded using the path to the exchange set's `CATALOG.031` file, and optionally the paths to `CATALOG.031` files in other exchange sets (in other folders) that contain additional update dataset files. If an exchange set that only contains update dataset files is one of those specified, then the exchange set with the corresponding base dataset files must be loaded simultaneously. See the topic section Working with updates for more information.

`EncDataset` objects represent datasets. Each `EncDataset` contains the base data for a cell and any updates that were also loaded when the exchange set was loaded, and provides access to metadata about the cell. Retrieve a collection of `EncDataset` objects contained in an `EncExchangeSet` using the `Datasets` property .

ENC cells (individual charts) are represented by `EncCell` objects. You can construct these objects in two ways:

- From a dataset represented by an `EncDataset` object. This approach will include the base dataset file and all corresponding update dataset files. This is the preferred approach.
- From a base dataset file. This approach will only include the base dataset file, and not any corresponding update dataset files. This is not a typical use case.

Copyright © 1995-2020 Esri. All rights reserved.

Display an ENC cell by constructing an `EncLayer` object from an `EncCell` object. `EncLayer` is derived from `Layer` , so you add an `EncLayer` to a map like you do other layers.

You can add all the charts in the exchange set by iterating through the exchange set's datasets, creating an `EncCell` for each, creating an `EncLayer` for each `EncCell` , and adding each `EncLayer` to the map.

```
EncExchangeSet encExchangeSet = new EncExchangeSet(new string[] { @"..\Data\ENC_ROOT\
CATALOG.031" });
await encExchangeSet.LoadAsync();

foreach(EncDataset encDataset in encExchangeSet.Datasets)
{
    EncLayer encLayer = new EncLayer(new EncCell(encDataset));
    MyMap.OperationalLayers.Add(encLayer);
}
```

Work with updates

ENC charts are often distributed as base cells (.000 files) with one or more update cells (.001, .002, etc. files). An exchange set can consist exclusively of update cells; in order to load an update exchange set, the path to the base exchange set must be provided along with the path to the update exchange set.

```
EncExchangeSet exchangeSet = new EncExchangeSet(new string[] { @"path\to\update", @"path\to\
original" });
```

When loading an ENC cell, it is important to use the dataset-based constructor. If updates for a cell are part of an exchange set, they will only be found by the runtime when the dataset-based constructor is used. Loading the cell from a path will not load any associated updates.

Set ENC environment settings

ENC layers are displayed in accordance with the IHO S-52 standard. You can define the display properties of your ENC layers by using the static `EncEnvironmentSettings` class. These settings apply to all ENC layers in all maps. Settings fall under three categories: mariner settings, text group visibility settings, and viewing group settings. Text group settings control the display of labels for features, mariner settings control the symbolization and presentation of ENC features, and viewing group settings allow for quickly applying settings to logical groups of feature types.

```
// Enables display of seabed information for all ENC layers
EncEnvironmentSettings.Default.EncDisplaySettings.TextGroupVisibilitySettings.NatureOfSeabed =
true;
```

ArcGIS Runtime’s ENC implementation works with ENC content via compiled **SENC** files. SENC is an acronym for System Electronic Navigational Chart. After an ENC cell has been loaded, all future loads of that cell will reference the underlying SENC file directly. You can use the `SencDataPath` property of the environment settings to find or set the location where SENC files are stored.

Copyright © 1995-2020 Esri. All rights reserved.

Identify and select ENC features

ENC layers support identify through the common geoview feature identification mechanism, which takes a screen point and tolerance in pixels. Identify will return a collection of `IdentifyLayerResult` objects, one per matching layer. For ENC layers, the results will have a `GeoElements` property, which is a collection of `EncFeatures`.

Once a feature has been identified, you can call `SelectFeature` on the layer that contains the feature to select it.

```
// Perform the identify operation
IReadOnlyList<IdentifyLayerResult> results = await MyMapView.IdentifyLayersAsync(e.Position, 5,
false);

// Return if there are no results
if (results.Count < 1) { return; }

// Take action on each identified ENC feature
foreach(IdentifyLayerResult result in results)
{
    foreach(EncFeature feature in result.GeoElements)
    {
        // Get the layer associated with this set of results
        EncLayer containingLayer = (EncLayer)result.LayerContent;

        // Select the feature
        containingLayer.SelectFeature(feature);
    }
}
```

Performance considerations

ArcGIS Runtime works with ENC content via internal SENC files. When an ENC cell is loaded, an SENC representation is generated. Subsequent loads only read the generated SENC files. When developing ArcGIS Runtime apps for working with ENC content, understand that:

- The SENC data path must be set before attempting to read ENC content. On .NET platforms, this is set to a temporary directory by default. The default directory may not be suitable for production use in the context of implementing an Electronic Chart Display and Information System (ECDIS).
- SENC files are device- and version-specific. These files should not be exposed to users, backed up, or shared between devices. The SENC format is internal to ArcGIS Runtime and may change between versions of ArcGIS Runtime. Therefore, SENC files created by an app built with one version of ArcGIS Runtime are not guaranteed to work with apps built with another version of ArcGIS Runtime, even on the same device. On iOS, consider setting the SENC path to a subdirectory of the documents directory that is excluded from iCloud and iTunes backup. See Apple Technical Q&A QA1719 for more information about excluding iOS directories from backup. See the Android Developers topic for more information about excluding directories from backup on Android.
- SENC files take time to generate—this will delay the loading of new ENC cells. It may take a long time to load large ENC exchange sets consisting of many cells - potentially hours. Never block the UI when loading these files.
- Because the initial load of an ENC cell (before the SENC files have been generated) can take some time to complete, consider pre-loading them to ensure availability before depending on them for navigation.
- Do not attempt to read or manipulate SENC files. Changes to generated SENC files will invalidate them, requiring ArcGIS Runtime to re-generate them the next time the corresponding cells are loaded.

Copyright © 1995-2020 Esri. All rights reserved.

Display symbols from a style with a dictionary renderer

When symbolizing geoelements in your map, you may need to convey several pieces of information with a single symbol. For example, you may want to symbolize restaurants so that each symbol reflects the type of food, price, rating, number of reviews, current health grade, seating capacity, average wait time, and so on. You can try to symbolize such data using a unique value renderer, but as the number of fields and values increases, that approach becomes impractical. With a dictionary renderer, however, you can build each symbol on the fly based on one or several attribute values and also handle a nearly infinite number of unique combinations.

Each component of a dictionary renderer's symbol is based on an attribute value and describes something about the geoelement it represents. This can be useful for data with attribute values that change frequently, since the symbol can update to show the current state of the geoelement. The following example shows a single symbol for a restaurant in which each component (symbol primitive) describes an aspect of the feature:

A dictionary renderer applies symbols to features through an associated dictionary symbol style. The style contains all the required symbols as well as (for newer format styles) logic and configurable properties for applying them. For details, see the How a dictionary renderer works section below.

Tip:

See the ArcGIS Pro documentation for information about using dictionary symbology inArcGIS Pro. There is also an open source dictionary renderer toolkit with tools, instructions, and examples to help you create a custom dictionary style for use in ArcGIS Pro or ArcGIS Runtime SDK apps.

Common tasks using a dictionary renderer and its associated style include the following:

- Symbolize a feature layer or graphics overlay
- Get a single symbol based on a set of input attributes

Symbolize a feature layer or graphics overlay with symbols from a dictionary style

1. Point to the dictionary style file (`.stylx`) and create a dictionary symbol style object.

   This can be one of the standard military styles supported with ArcGIS Runtime SDK, or a custom dictionary style. Custom styles use the newer (Arcade-based) styles that have metadata with the specification name. The `CreateFromFileAsync` constructor is used for newer Arcade-based styles and will fail if you provide an older style. The `OpenAsync` constructor, deprecated at 100.6.0, can be used exclusively for older styles. When opening an older style, you must also provide the name of the style specification.

```
// Open a custom Arcade-based style using a path to the .stylx file.
DictionarySymbolStyle restaurantStyle = await
DictionarySymbolStyle.CreateFromFileAsync(stylefilePath);

// Open an older non-Arcade style by specifying the specification name and path to the .stylx
```

Copyright © 1995-2020 Esri. All rights reserved.

```
file.
DictionarySymbolStyle militaryStyle = await DictionarySymbolStyle.OpenAsync("mil2525d",
_styleFilePath);
```

2. ArcGIS Runtime will automatically discover input fields whose names match the dictionary style's expected fields for symbols and text. However, if certain field names don't match (or you want to use different fields), you can explicitly map the expected fields to the appropriate fields in the data.
   Field mapping overrides are defined with two sets of key-value pairs: one for symbol fields and one for text fields. Each key identifies an expected field (defined in the dictionary's symbol and text properties) and the value identifies the corresponding mapped field (from the dataset). Field names are not case sensitive.

```
// Create overrides for expected field names that are different in this dataset.
Dictionary<string, string> styleToFieldMappingOverrides = new Dictionary<string, string>
        {
                { "style", "foodtype" },
                { "healthgrade", "inspection_score" },
                { "rating", "stars"},
                { "price", "price_range"},
                { "opentime", "opens" }
        };

// Create overrides for expected text field names.
Dictionary<string, string> textFieldOverrides = new Dictionary<string, string>
        {
                { "name", "RestaurantName" }
        };
```

3. Set configuration property values for the style. Configuration properties are stored in a read-only list of symbol style configuration objects. Access the desired configuration in the list and change its value. This step is only necessary if you want to change any of the default configuration settings.

```
// Update a configuration settings for the style.
restaurantStyle.Configurations.ToList().Find(config => config.Name == "text").Value = "ON";
```

4. Create a dictionary renderer that uses the style. If you need to use custom field mappings (step 2), include the key-value pairs that define them.

```
// Create the dictionary renderer with the style file and the field overrides.
DictionaryRenderer dictRenderr = new DictionaryRenderer(restaurantStyle,
styleToFieldMappingOverrides, textFieldOverrides);
```

5. Assign the dictionary renderer to the layer or graphics overlay.

```
restaurantLayer.Renderer = dictRenderr;
```

Copyright © 1995-2020 Esri. All rights reserved.

Get a symbol from a dictionary style for a specified set of attributes

1. Point to the dictionary style file (`.stylx`) and create a dictionary symbol style object.

```
// Open a custom Arcade-based style using a path to the .stylx file.
DictionarySymbolStyle restaurantStyle = await
DictionarySymbolStyle.CreateFromFileAsync(restaurantStyleFilePath);
```

2. Define a set of attributes with which to build the symbol.
   These are key-value pairs containing the expected attribute name and the input value.

```
Dictionary<string, object> attributes = new Dictionary<string, object>
{
    { "style", foodStyle },
    { "healthgrade", inspection_score },
    { "rating", numberOfStars},
    { "price", priceRange},
    { "opentime", opensAt },
    { "closetime", closesAt }
};
```

3. Use the attribute values to search the style and return the appropriate multilayer symbol.

```
Symbol restaurantSymbol = await restaurantStyle.GetSymbolAsync(attributes);
```

Note:
You can also pass a set of keys (instead of attributes) to return a symbol. See Read symbols from a style file for additional ways to retrieve symbols from a style.

How a dictionary renderer works

A renderer is a component that contains a collection of (one or more) symbols and some logic that determines how those symbols are applied to geoelements in a layer or graphics overlay. ArcGIS Runtime SDK provides several types of renderers for displaying symbols and they are described in Symbolize data.

A dictionary renderer applies symbols from an associated style file, stored in an SQLite database with a `.stylx` extension. Each symbol in the style is identified with a unique key. A style can be opened in ArcGIS Pro, and symbols can be added, deleted, or modified.

Copyright © 1995-2020 Esri. All rights reserved.

For versions prior to 100.6.0, a dictionary renderer was used exclusively for displaying military symbols. The logic (known as the rule engine) for displaying the symbols was built into ArcGIS Runtime binaries, required a style based on one of the supported specifications, and was not open to customization. Starting with ArcGIS Pro 2.4 (and supported in ArcGIS Runtime SDK 100.6.0), the logic for applying symbols from a dictionary style is implemented as an Arcade script and stored in the style file along with the symbols. This allows you to create your own dictionary style with the symbols you need as well as the logic for how they are applied. A JSON definition of dictionary configuration properties (also stored in the style) allows you to define the expected attributes and other settings used by the style.

Note:
Older versions of military styles (those that don't use Arcade) are still supported but use of the new format is encouraged when possible.

One or more symbols are read from a dictionary style to build a composite multilayer symbol for a geoelement. A dictionary renderer determines the symbol components to request for a geoelement's symbol (using symbol keys) based on input attribute values and style-specific logic. The Arcade code below, for example, reads the input value for the health inspection (`$healthgrade`), determines the corresponding letter grade, and stores the appropriate key to return.

```
if (!isempty($healthgrade) && $healthgrade > 0) {
    if($healthgrade > 89){ healthgradekey = 'inspection-A' }
    else if($healthgrade > 79){ healthgradekey = 'inspection-B' }
    else if($healthgrade > 69){ healthgradekey = 'inspection-C' }
    else { healthgradekey = 'inspection-yuk' }
  }
```

While you can open a dictionary style in ArcGIS Pro for editing symbols, you cannot access the Arcade logic or configuration there. You can use a utility for opening SQLite databases (such as DB Browser for SQLite), however, to view and edit those properties of the style.

Copyright © 1995-2020 Esri. All rights reserved.

For more information about creating your own dictionary style, see the Dictionary Renderer Toolkit.

Dictionary style properties

Dictionary style properties include symbol properties, text properties, and configuration properties. Symbol and text properties define the attribute fields expected by the style to display symbols and text. Configuration properties provide settings to control specific aspects of their display. When the renderer is applied, expected attribute names in the symbol and text properties are automatically mapped to fields in the data that have a matching name. To use an input field that doesn't match an expected attribute, you can explicitly map the input field to one of the configured attribute names in the symbol or text properties. This allows more flexibility for applying a style to datasets that have different names for input fields or that have several fields with appropriate input values.

Note:
Automatic matching of input fields to expected fields in the dictionary properties in not case sensitive. The field "IDENTITY" will automatically match a configured field called "identity", for example.

The following example shows the configuration JSON for the restaurant dictionary style illustrated previously. In this case, there is only one configuration property, which is the ability to turn text display on or off. Symbol properties are defined with a list of the expected attributes for creating geoelement symbols. The text properties list contains attributes used to display text with the symbol (in this case, only a "name" field).

```
{
  "configuration": [{
      "name": "text",
      "value": "ON",
      "domain": ["ON", "OFF"],
      "info": "indicates if the text is rendered"
    }
  ],
  "symbol": ["style", "rating", "price", "healthgrade", "opentime", "closetime"],
  "text": ["name"]
}
```

Text can be displayed as additional symbol components using the values contained in the specified field or fields. The configuration property for showing or hiding text can be used to turn off all text display, regardless of the input attribute values.

Copyright © 1995-2020 Esri. All rights reserved.

Map geoelement fields to configuration fields

The dictionary renderer will automatically read fields in your data that have names matching those required by the style specification. For any fields in your data that don't exactly match the expected names, map the field names by setting the `SymbologyFieldOverides` and `TextFieldOverrides` properties of the dictionary renderer. These operations take a set of key-value pairs, in which the key is the expected attribute name (for example, healthgrade), and the value is the corresponding attribute name from the dataset (for example, inspection_score).

If you're not sure what attributes are defined in the symbol and text configuration properties, you can obtain a string list of the expected symbology and text fields by calling `SymbologyFieldNames` and `TextFieldNames` on `DictionarySymbolStyle`.

Military symbols

Military symbols are based on military specifications (military standards, such as MIL-STD-2525C and MIL-STD-2525D) and are composed of many elements such as frames, icons, modifiers, graphic amplifiers, and text amplifiers. In ArcGIS, military symbols are composed of multiple symbol layers. The ArcGIS Solutions for Defense team hosts military symbology styles for the following standards: MIL-STD-2525D, MIL-STD-2525C, MIL-STD-2525B w/CHANGE 2, APP-6(B), and APP-6(D). Use the support matrix to find the supported stylx file for your version of ArcGIS Runtime SDK.

Military symbol dictionary styles allow you to choose whether to assemble and render the symbol based on a single attribute with a unique Symbol ID Code (SIC or SIDC) or based on a series of predefined attributes. For example, in MIL-STD-2525BC2, a SIC of `SFSPCL---------` represents this symbol . The ArcGIS Military Overlay is a solution for creating and sharing military overlays in ArcGIS Pro according to a specific schema (such as identity, symbol set, and so on), each having a range of valid values associated with it. This schema will work by default with the military symbol dictionary style.

Regardless of the specification, all military symbols are based on a varying combination of attributes or codes and a set of rules about how the symbols should display.

Copyright © 1995-2020 Esri. All rights reserved.

Search

Copyright © 1995-2020 Esri. All rights reserved.

Identify features

Maps and scenes often combine multiple sources of information such as feature layers, map image layers, raster layers, and graphics. Labels and legends don't always provide enough information to work out what the map or scene is displaying. Use the identify methods to quickly answer the question: what is this item here? This will allow your users to explore and learn about the map or scene content by tapping or clicking it. Information returned can be shown in pop-ups or other UI components in your app.

You can identify the visible items at a specific point on screen, such as the following:

- Within all the layers in the map or scene, or only within a specific layer
- Within all the graphics overlays in the view or only within a specific graphics overlay
- Returning only the topmost item or all the items at that location
- Returning the feature, graphic, raster cell value, or other item at that location, or by returning pop-ups for pop-up-enabled layers

Identify methods are asynchronous, so that the UI thread of your application is not blocked waiting for results. This is especially important when working with data that resides on a server, as results do not return immediately. The results of an identify take into account the following:

- Symbology—Tapping on a large marker symbol used to draw a point, or a wide line symbol used to draw a polyline or outline of a polygon will include those items in the results; it's not just the geometry that is used.
- Visibility—The visibility of a graphics overlay or layer, and of an individual feature or graphic is checked, and results will only include the visible items. The opacity of a layer or graphic is ignored.
- Visible extent—Only items within the currently visible map or scene extent are included in identify results.

Identify is supported on feature layers, map image layers, Web Map Service (WMS) layers, graphics overlays, scene layers, raster layers, and also on tiled layers that are based on map services that support identify.

Note:
If time-based filtering is being used (a time extent has been set on the displaying map or scene view), identify will only return features that are within the time extent set on the geo view.

The sections below show you how to identify different items in the map or scene in different ways, but these approaches can be combined to provide general identify functionality if required.

Identify features in a feature layer

The steps below show you how to identify the features within a specific feature layer in the map or scene. Later in this topic, these steps are adapted to identify only the topmost feature, identify against multiple layers, different types of layers, and to identify graphics.

1. Listen to a tap or click event on the map or scene view, and get the point representing the center of the tap or click.

```
// handle the tap event on the map view (or scene view)
MyMapView.GeoViewTapped += async (s, e) =>
{
    // get the tap location in screen units
    System.Windows.Point tapScreenPoint = e.Position;
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    // ... code here to identify using the tap point ...
};
```

2. Call the required identify method, passing in the screen point from the previous step.
   a. Choose either to identify against a specific layer, or to identify against all layers.
   b. Specify a tolerance for the search radius of the identify operation. A tolerance of 0 identifies only items at the single pixel at the screen point. However, typically this level of precision is hard to achieve, so you can supply a tolerance around the screen point. A suitable tolerance for a tap operation on a touch screen should be equivalent to the size of the tip of a finger, and a smaller tolerance should be considered for mouse pointers.
   c. Specify whether to include pop-up information on the geo-elements returned. (If no pop-ups are defined for a layer or graphics overlay, pop-ups will not be included in the results, regardless of this parameter value.)
   d. Optionally, specify the maximum number of results per layer to return. This may be especially useful if identifying on service layers, as you can limit the amount of information returned to the client (the maximum results will also be limited by the server).

```
var layer = MyMapView.Map.OperationalLayers[0];
var pixelTolerance = 20;
var returnPopupsOnly = false;
var maxResults = 5;

// identify a layer using MapView, passing in the layer, the tap point, tolerance, types to
return, and max results
IdentifyLayerResult idResults = await MyMapView.IdentifyLayerAsync(layer, tapScreenPoint,
pixelTolerance, returnPopupsOnly, maxResults);
```

3. Results consist of layer content information about the layer the results are from, and a list of `GeoElement`. Each `GeoElement` represents a feature identified at the given screen point. Iterate the results, access the geometry and attributes of the identified features, and use them as required. In the example below, each identified feature is selected.

```
// get the layer identified and cast it to FeatureLayer
FeatureLayer idLayer = idResults.LayerContent as FeatureLayer;

// iterate each identified GeoElement in the results
foreach (GeoElement idElement in idResults.GeoElements)
{
    // cast the result GeoElement to Feature
    Feature idFeature = idElement as Feature;

    // select this feature in the feature layer
    idLayer.SelectFeature(idFeature);
}
```

Tip:

Features are loadable, but when using identify methods, they are always returned already loaded.

Copyright © 1995-2020 Esri. All rights reserved.

Identify features in all feature layers

When you do not know which specific layer to identify on, you can identify items in any layer in the map or scene. Change the first workflow above so that you do not specify the layer to identify. This time, the results are returned as a list of `IdentifyLayerResult` instead of a single result.

In the following example, the results are iterated to select identified features from all layers in the result list:

```
// get the tap location in screen units
System.Windows.Point tapScreenPoint = e.Position;

var pixelTolerance = 20;
var returnPopupsOnly = false;
var maxLayerResults = 5;

// identify all layers in the MapView, passing the tap point, tolerance, types to return, and
max results
IReadOnlyList<IdentifyLayerResult> idLayerResults = await
MyMapView.IdentifyLayersAsync(tapScreenPoint, pixelTolerance, returnPopupsOnly, maxLayerResults);

// iterate the results for each layer
foreach (IdentifyLayerResult idResults in idLayerResults)
{
    // get the layer identified and cast it to FeatureLayer
    FeatureLayer idLayer = idResults.LayerContent as FeatureLayer;

    // iterate each identified GeoElement in the results for this layer
    foreach (GeoElement idElement in idResults.GeoElements)
    {
        // cast the result GeoElement to Feature
        Feature idFeature = idElement as Feature;

        // select this feature in the feature layer
        idLayer.SelectFeature(idFeature);
    }
}
```

Layers that do not support identify or do not have any results based on the inputs are not included in the returned list.

Identify topmost item only

To identify only the topmost item at the screen point (or the topmost item per layer if identifying against all layers), change the first workflow above by removing the maximum number of results parameter. `GeoElements` are still returned as a list, but the list will have only zero or one items.

The following code selects only the topmost feature identified at the screen point, for all feature layers in the map.

```
// get the tap location in screen units
System.Windows.Point tapScreenPoint = e.Position;

var pixelTolerance = 20;
var returnPopupsOnly = false;

// identify the topmost feature for all layers, passing in the tap point, tolerance, and types
to return
IReadOnlyList<IdentifyLayerResult> idLayerResults = await
MyMapView.IdentifyLayersAsync(tapScreenPoint, pixelTolerance, returnPopupsOnly);

// iterate the results for each layer
foreach (IdentifyLayerResult idResults in idLayerResults)
```

Copyright © 1995-2020 Esri. All rights reserved.

```
{
    // get the layer identified and cast it to FeatureLayer
    FeatureLayer idLayer = idResults.LayerContent as FeatureLayer;

    // see if a result was found for this layer (will be 0 or 1 result for each layer)
    if (idResults.GeoElements.Count == 1)
    {
        // cast the result GeoElement to Feature
        Feature idFeature = idResults.GeoElements[0] as Feature;

        // select this feature in the feature layer
        idLayer.SelectFeature(idFeature);
    }
}
```

Identify on group layers

The results returned when any group layers are in the map or scene depends on the identify operation you use.

When identifying on the map or scene using `IdentifyLayersAsync` (plural layers), identification operates on all the child layers of any group layers independently of the group layer. There is no result for the group layer itself, only for the child layers. Effectively, this behavior treats the child layers as independent layers and otherwise ignores the group layers.

When identifying specifically on the group layer using `IdentifyLayerAsync` (singular layer), identification operates on all the child layers of the group layer. The result is returned in an `IdentifyLayerResult` object for the group layer. `IdentifyLayerResult.GeoElements` will not contain any results for the group layer itself, so instead, access results for each child layer through `IdentifyLayerResult.SublayerResults`. This returns a list of `IdentifyLayerResult` objects for each child layer with the elements identified in each layer.

```
// Specify identify properties.
double pixelTolerance = 20.0;
bool returnPopupsOnly = false;
int maxResults = 5;

// Identify a  group layer using MapView, passing in the layer, the tap point, tolerance, types
to return, and max results.
IdentifyLayerResult groupLayerResult = await MyMapView.IdentifyLayerAsync(groupLayer,
tapScreenPoint, pixelTolerance, returnPopupsOnly, maxResults);

// Iterate each set of child layer results.
foreach (IdentifyLayerResult subLayerResult in groupLayerResult.SublayerResults)
{
    // Display the name of the sublayer.
    Console.WriteLine("Results for child layer: " + subLayerResult.LayerContent.Name);

    // Iterate each geoelement in the child layer result set.
    foreach (var idElement in subLayerResult.GeoElements)
    {
        // Loop through and display the attribute values.
        Console.WriteLine("  ** Attributes **");
        foreach(KeyValuePair<string, object> attribute in idElement.Attributes)
        {
            Console.WriteLine("    - " + attribute.Key + " = " + attribute.Value.ToString());
        }
    }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Identify on map image layers

Identifying against map image layers and tiled map layers follows the same workflow as shown for feature layers. The following are differences when identifying against map image layers:

- Results are returned as `Features`; unlike other features, however, they will not have a reference to a `FeatureTable`.
- Map image layers may have one or more sublayers—identify results from map image layers reflect this structure, and return results for each sublayer separately. (Note that if you have specified a maximum number of results to return, this value applies per sublayer.)

The following code processes identify results for feature layers and map image layers. A recursive function is used to iterate through sublayer results for map image layers.

```
private void ProcessIdentifyResults(IReadOnlyList<IdentifyLayerResult> idLayerResults)
{
    // iterate the results for each layer
    foreach (IdentifyLayerResult idResults in idLayerResults)
    {
        // get the layer identified and try to cast it to FeatureLayer
        FeatureLayer featLayer = idResults.LayerContent as FeatureLayer;

        // process feature layer results
        if (featLayer != null)
        {
            // iterate each identified GeoElement in the results for this layer
            foreach (GeoElement idElement in idResults.GeoElements)
            {
                // cast the result GeoElement to Feature
                Feature idFeature = idElement as Feature;

                // select this feature in the feature layer
                featLayer.SelectFeature(idFeature);
            }
        }

        // if not a feature layer, check for map image layer
        if (idResults.LayerContent is ArcGISMapImageLayer || idResults.LayerContent is
ArcGISMapImageSublayer)
        {
            // iterate each identified GeoElement in the results
            foreach (GeoElement idElement in idResults.GeoElements)
            {
                // cast the result GeoElement to Feature
                Feature idFeature = idElement as Feature;

                // can get the geometry and attributes, but FeatureTable will be null
                Geometry featureShape = idFeature.Geometry;
                IDictionary<string, object> featureAttrs = idFeature.Attributes;

                // ... code to display the GeoElement attributes here ...

            }

            // see if there are sublayer results, call this function again and pass them in
            ProcessIdentifyResults(idResults.SublayerResults);
        }
    }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Return pop-ups as results

Some types of layers support pop-ups. You can use an identify method to return pop-ups for map content at a screen point.

The following code provides an example of getting a pop-up from the identify results:

```
// identify all layers in the MapView, passing the tap point, tolerance, types to return, and
max results
IReadOnlyList<IdentifyLayerResult> idLayerResults = await
MyMapView.IdentifyLayersAsync(tapScreenPoint, pixelTolerance, returnPopupsOnly, maxLayerResults);
  if (idLayerResults.Count > 0)
{
    // get the first identify result
    IdentifyLayerResult firstIdLayerResult = idLayerResults.FirstOrDefault();

    if (firstIdLayerResult.Popups.Count > 0)
    {
        // get the first popup from the result
        Popup firstPopup = firstIdLayerResult.Popups.FirstOrDefault();

        // ... code here to show popup ...
    }
}
```

Identify features in a WMS layer

WMS layers differ from other layers, as they do not support returning individual attributes or geometry for a feature. WMS services perform identification on the server and return HTML documents describing identified features. You can access the returned HTML document string through the "HTML" entry in the feature's attributes. This HTML string is suitable for display in a web view.

It is impossible to get the geometry for an identified (or any other) WMS feature. An identified WMS feature's geometry will always be null. Consequently, WMS layers do not support feature selection/highlight.

```
// Perform the identify operation on the WMS layer
IdentifyLayerResult myIdentifyResult = await MyMapView.IdentifyLayerAsync(myWmsLayer,
e.Position, 20, false);

// Return if there's no feature to show
if (myIdentifyResult.GeoElements.Count() < 1) { return; }

// Retrieve the identified feature, which is always a WmsFeature for WMS layers
WmsFeature identifiedFeature = (WmsFeature)myIdentifyResult.GeoElements[0];

// Retrieve the WmsFeature's HTML content
string htmlContent = identifiedFeature.Attributes["HTML"].ToString();
```

Identify graphics

Graphics are identified using methods different than those used for layers. You can choose to return graphics in a specific graphic overlay or for all graphics overlays; you can also limit results to only the topmost graphic in each graphics overlay.

The following code identifies all graphics overlays in the map view or scene view, then loops through all graphics and selects them:

```
// identify all graphics overlays in the MapView, passing the tap point, tolerance, types to
return, and max results
IReadOnlyList<IdentifyGraphicsOverlayResult> idGraphicOverlayResults = await
```

Copyright © 1995-2020 Esri. All rights reserved.

```
MyMapView.IdentifyGraphicsOverlaysAsync(tapScreenPoint, pixelTolerance, returnPopupsOnly,
maxLayerResults);

// iterate each graphics overlay
foreach(IdentifyGraphicsOverlayResult idGraphicResult in idGraphicOverlayResults)
{
    // iterate all graphics in the overlay and select them
    foreach(Graphic g in idGraphicResult.Graphics)
    {
        g.IsSelected = true;
    }
}
```

Identify on raster layers

Identify on raster layers returns the `RasterCell` value for a tapped location in a `MapView` or `SceneView`. The identified `RasterLayer` can be local on the device or from a web service layer hosted on ArcGIS Online or Portal for ArcGIS. Identify returns raster cell values to display in a simple callout, or if a pop-up is configured for the raster layer, the information can be displayed in a formatted UI.

The following code calls `MapView.IdentifyLayersAsync` to return identify results for layers in the map. Since the results might be from a variety of layers types, each `GeoElement` result is first cast to `RasterCell` to ensure only raster results are processed.

```
// Identify layers at the tap location
IReadOnlyList<IdentifyLayerResult> identifyResults = await
MyMapView.IdentifyLayersAsync(e.Position, 1, false, 10);

// Loop through the results for all layers
foreach (IdentifyLayerResult layerResult in identifyResults)
{
    // Get the readonly list of geo-elements
    // Depending on the type of layer identified, these can be Graphics, Features, or RasterCells
    IReadOnlyList<GeoElement> resultElements = layerResult.GeoElements;

    // Create a StringBuilder to display information to the user
    StringBuilder myStringBuilder = new StringBuilder();

    // Loop through each result element, only process raster cells
    foreach (GeoElement element in resultElements)
    {
        // Cast the result GeoElement to a RasterCell, return if it's null
        RasterCell cell = element as RasterCell;
        if (cell == null) { return; }

        // Loop through the attributes (key/value pairs)
        foreach (var keyValuePair in cell.Attributes)
        {
            // Add the key/value pair to the string builder
            myStringBuilder.AppendLine(keyValuePair.Key + ": " + keyValuePair.Value);
        }

        // Get the X/Y coordinates for the cell center
        double theX = cell.Geometry.Extent.GetCenter().X;
        double theY = cell.Geometry.Extent.GetCenter().Y;

        // Format the X/Y values as a human readable string
        string coordString = "X: " + Math.Round(theX, 4) + " Y: " + Math.Round(theY, 4);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
        // Add the X/Y cell coordinates to the string builder
        myStringBuilder.AppendLine(coordString);

        // Define a callout based on the string builder
        CalloutDefinition myCalloutDefinition = new
CalloutDefinition(myStringBuilder.ToString());

        // Display the callout in the map view
        MyMapView.ShowCalloutAt(e.Location, myCalloutDefinition);
    }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Search for places (geocoding)

Using a place name or street address as input, your user can find locations of interest and interact with them on the map. The process of matching locations on the map to an address is referred to as geocoding.

Geocoding overview

Address geocoding (or geocoding) is the process of using information contained in an address to interpolate a corresponding location on the map. An address may define a precise street location, such as `380 New York Street, Redlands CA 92373`. It may provide only a general location, such as a city name or postal code, or it may be a familiar place name or landmark, such as `Ayers Rock`.

Using a reference data source, referred to as a locator, ArcGIS finds map locations by matching an input address with feature attributes. The number and accuracy of the geocoded results (often called matches or candidates) may vary depending on the specificity and quality of the input address provided. For example, the address `New York` is likely to return a match for the state of New York, the city of New York, and perhaps several streets or points of interest that share that name. Geocoded locations are always points. If a match is made with a polygon (a state or postal code, for example), the match location represents the center of that area.

Additional geocoding inputs can improve the relevance of results by restricting the search to a given area or finding matches that are near a reference location. This allows you to find results for a specified country, map extent, or near the user's current or anticipated location. A general search such as `restaurant` is unlikely to return useful candidates unless it is restricted to a specific area.

When several candidates are returned for an address, they are sorted by the quality of the match. This means it is generally safe to assume that the first candidate in the results represents the best match. Sometimes, however, you may want to examine a set of top candidates to determine the most appropriate match. Address candidates can contain the level at which they were matched (house, street, postal code, and so on) and the interpolated address created by the locator. Both of these may help you determine the best match or perhaps find problems in the original address input. If geocoding used a reference location, a distance attribute can help you find the closest candidate.

⚠ Caution:

The score attribute, an integer value ranging from 0 to 100, provides an estimate of relative match accuracy within a single locator. If you are using a composite locator (such as the ArcGIS World Geocoding Service), these values will represent the score within a locator for a specific level, but not an overall evaluation of the match. A score of 100 for a postal code match is not necessarily a more precise result, for example, than a score of 90 at the house level.

Addresses

Addresses are fundamental for geocoding and have some specific characteristics. An address is composed of one or more address elements: individual components of the address, such as house number, street name, street type, and postal code. Address elements help in the geocoding search, pinpointing an address to a particular location. Addresses come in various styles and formats, including street address, a place name, or a location that is identified by a code. A common address format used in the United States consists of the following series of address elements: house number, prefix direction, prefix type, street name, street type, suffix direction, and zone information (such as city, state, and ZIP Code). Globally, addresses are represented in a variety of formats. However, while all of these addresses differ to some degree, some things remain consistent. Each address consists of one or more address elements presented in a particular address format recognized by those in the region.

Copyright © 1995-2020 Esri. All rights reserved.

When geocoding, you can pass in individual components of an address or provide all information as a single string to be parsed. Defining each component of an address gives you more control but may require that you either parse the components before passing them to the locator or provide additional UI controls for your user to enter each piece of information.

When geocoding, the type of information expected in an input address is determined by the address style configured by the locator you're using. Some of the common address component field names for the ArcGIS World Geocoding Service (`https://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer`) are listed below as examples of the types of input that may be expected. See findAddressCandidates for details.

- Address—House number and street
- Neighborhood—Subdivision of a city (not used for United States addresses)
- City—City or municipality
- Subregion—Administrative region, such as a county or province (not used for United States or Mexico addresses)
- Region—Largest administrative boundary associated with the address (state name in the United States)
- Postal—Postal code
- CountryCode—Name or ISO code of the country

Locators

The fundamental logic for geocoding is built into the locator and does not come from your ArcGIS Runtime SDK code. The locator (created using ArcGIS Desktop) is the major component in the geocoding process and contains all the data necessary to perform address matching. If you'd like a ready-to-use and regularly updated locator (and network dataset) for your area of interest, you can license StreetMap Premium data (in MMPK format). For details, see Add StreetMap Premium data.

A locator is created based on a specific address locator style, which dictates what type of address input is expected and how matching will be carried out. The following table lists some of the common address styles, the type of matching that it enables, and an example of input address values:

Address style	Description	Input address examples
US Address - City State	Finds a city in the United States using reference data that stores city and state names	Bonita, CA Punxsutawney, PA
US Address - Dual Ranges	Interpolates an address location using a streets reference dataset that includes number ranges for both sides of the street	30 E. State St, Redlands CA 92373 1060 W. Addison St, Chicago IL 60613
General - Gazetteer	Finds place names or landmarks worldwide using reference data with a place name attribute	Taj Mahal Ypres
General - Single Field	Locates features using a specific field in the reference dataset	012883902 UTC330868-A

Note:
This is not a complete list of the available address styles. For a more complete description, see Commonly used address locator styles in the ArcGIS Desktop help.

Copyright © 1995-2020 Esri. All rights reserved.

Once created, an address locator contains a set of geocoding properties (preferences), a snapshot of the address attributes in the reference dataset, and the queries for performing a geocoding search. The address locator also contains a set of address parsing and matching rules that directs the geocoding engine to perform address standardization and matching.

A locator can be stored locally (as a *.loc file) or can be published as a locator service using ArcGIS Server. To perform geocoding, you need to access either an online locator or one available locally on the device. Using ArcGIS Runtime SDK, you can connect to a variety of online locators published and hosted by Esri, including address locators for Europe and North America, a street locator for the United States, and a world places locator. These services allow you to begin geocoding with online data right away without having to create and configure an address locator yourself. Locators can be created and configured with local data using ArcGIS Desktop as the authoring environment. See Creating ArcGIS Runtime content for detailed information.

Note:
Without the *.locb file, ArcGIS Runtime SDK is not able to use the locator in an offline environment.

Results

Geocoding results for a particular address are referred to as candidates. Depending on how specific and complete the input address is, you may get several candidates from a geocode operation. Geocoding results are ordered by quality of the match, so the first candidate is generally the best. Additional candidate information can be obtained by specifying supplemental output fields to include in the results. The following table describes a few of the available fields you can include from the world geocoding service. This is only a representative list of fields. For a complete list, see Output fields in the ArcGIS REST API documentation.

Note:
The fields provided may vary between locators. See the Get locator info section of this topic for information about querying a locator for its available result attributes.

Output field	Description
Score	Value between 0 and 100 that describes the relative quality of the match within a single locator.
Match_addr	Complete address returned for the geocode request. The format is based on address standards for the country within which the address is located.
Addr_type	The match level for a geocode request. For example, a candidate that matches only to a postal code will have a value of `Postal`. Supported match levels vary in different countries. See Output fields for a description of possible values for this field.
Side	The side of the street on which the address was matched (relative to the direction of digitization of the reference line feature). Possible values are `L` (left), `R` (right), or an empty string (undefined).
Distance	The distance (meters) from the candidate to a specified location. The reference location needs to be specified as a parameter before executing the task.
Rank	A number indicating the importance of each candidate relative to others in the results. Rank is generally used to help distinguish ambiguous place names using population as the criterion. A search for `Las Vegas`, for example, may return `Las Vegas, NV` as the top ranked match (1) and `Las Vegas, NM` as a lower rank based on the relative population of those cities. If a reference location and distance parameters have been set for a find operation, distance from the reference location will be used to rank candidates.

Copyright © 1995-2020 Esri. All rights reserved.

Use a locator task

Geocoding is implemented using the `LocatorTask` class with an online or local locator. To implement this task in your code, follow these general steps:

1. Create a `LocatorTask` using an online or local resource (locator service or file).
2. Specify geocoding preferences using the `GeocodeParameters` object. This is optional but allows you to request additional attributes in the results, to restrict the search by area, set the maximum number of results, and so on.
3. Execute the task, passing in any required parameters (such as address values and geocode parameters).
4. Process the results (display a list of candidates for the user to choose from, display the best match as a graphic on the map, and so on).

You create a `LocatorTask` by passing a URI (for an online locator) or a path (for a local locator) to the static `CreateAsync` method on the class. The following example creates a new `LocatorTask` that uses the ArcGIS Online World Geocoding service.

```
var geocodeServiceUrl = @"http://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer";
LocatorTask geocodeTask = await LocatorTask.CreateAsync(new Uri(geocodeServiceUrl));
```

Note:

The World Geocoding Service from ArcGIS Online includes a `for_storage` parameter. Storing the results of a geocode or reverse geocode operation using this service (by setting the `for_storage` parameter to **true**) requires an ArcGIS Online paid subscription and uses credits from the subscriber's account. Additionally, the user performing a paid operation must be assigned a user role that includes the geocoding privilege. For more information, see Free vs. paid operations in the REST documentation for this geocode service and User types, roles, and privileges in ArcGIS Online Help.

Get locator info

Before executing a geocode operation, you may need information about a particular locator to work with it effectively. You may need to know which input fields are used to locate an address, which attributes are available with the results, or the spatial reference in which the candidate locations will be returned by default. You can use `LocatorInfo` to get the following information about a locator:

- Name—A string that identifies the locator
- Description—A brief description
- Result attributes—A list of field names that are available for results
- Search attributes—A list of input fields used for multiline geocoding
- Spatial reference—The spatial reference used by the locator (and the default coordinate system for results)
- Support for intersections—A true or false value that indicates whether geocoding of street intersections is supported
- Support for points of interest—A true or false value that indicates whether searching points of interest (POI) is supported
- Support for suggestions—A true or false value that indicates whether the locator supports offering suggestions for addresses
- Version—The version of the locator

The following example shows how you can use a `LocatorTask` to connect to an online locator service and get the names of available result attributes:

Copyright © 1995-2020 Esri. All rights reserved.

```
// get info from the LocatorTask
var locatorInfo = geocodeTask.LocatorInfo;

// loop through all attributes available to include in the results
foreach(var field in locatorInfo.ResultAttributes)
{
    // add the field's display name to a list box
    ResultFieldsList.Items.Add(field.DisplayName);
}
```

Geocode parameters

When geocoding an address or point of interest, you can optionally provide preferences to control certain aspects of the geocoding operation. The following list describes some of the properties you can control:

- Country—The full name or code for the country to which geocoding should be restricted
- Maximum results—A limit for the maximum number of address candidates to return
- Output language—The language (culture) in which result attributes will be returned
- Output spatial reference—The spatial reference in which geographic locations for address candidates will be returned
- Result attributes—A list of field names to include in the results (available fields can be queried from the locator)
- Search area—A geographic shape that defines an area to geocode within

A similar (more limited) set of preferences is available when getting a set of address suggestions, as described in the following section.

Get address suggestions

You can use a locator task to find suggestions for a partial address entry. As your user types an address, for example, a list of suggestions based on the first few characters of the address string can be displayed. Check `LocatorInfo.SupportsSuggestions` to see if a particular locator can provide address suggestions.

A search for address suggestions can optionally be restricted to a specified country or area on the map. To define a country, use the full name or ISO 3166-1 (two or three character) code. See Geocode coverage in the REST API documentation for a list of countries and corresponding codes. You can also indicate a preferred location (point) for the search, and set the maximum number of suggestions to return.

Copyright © 1995-2020 Esri. All rights reserved.

Tip:

Restricting your search or geocode operations to a specified country or location (as well as the number of results) can improve performance.

The following example calls the `SuggestAsync` method on a `LocatorTask` to get suggested addresses as the user types. A `SuggestParameters` object is used to restrict the search for suggestions to the current map extent.

```
// get suggestions if the locator supports them
if (_geocodeTask.LocatorInfo.SupportsSuggestions)
{
    // get the current map extent
    var currentExtent =
MyMapView.GetCurrentViewpoint(ViewpointType.BoundingGeometry).TargetGeometry;
    // restrict the search to the current map extent and return no more than 10 suggestions
    var suggestParams = new SuggestParameters { MaxResults = 10, SearchArea = currentExtent };
    // get suggestions for the text provided by the user
    var suggestions = await _geocodeTask.SuggestAsync(AddressTextBox.Text, suggestParams);
    // show suggestions ...
    SuggestionsListBox.ItemsSource = suggestions;
}
```

The address suggestion results do not contain location information, only the full address text. To get a location, you must take the additional step of geocoding the suggestion as described in the following section.

Geocode an address

Use a `LocatorTask` to find geocode candidates from an address, point of interest, or address suggestion. An input address can be defined as a single string, such as `380 New York Street, Redlands CA 92373` or you can define the individual components of the address (such as number and street, city, region, postal code, and so on).

The following example defines an input address using a dictionary to hold each address component. Match candidates are then found by passing it to the `GeocodeAsync` method.

```
// create a dictionary of strings
var address = new Dictionary<string, string>();
// add each address component (keys and values)
address.Add("Address", "380 New York Street");
address.Add("City", "Redlands");
address.Add("Region", "CA");
address.Add("Postal", "92373");

// find matches for this address
var matches = await _geocodeTask.GeocodeAsync(address);
```

Tip:

You can use `LocatorInfo` to get the input address field names for a locator.

If you define the address as a single string, it will be parsed into the appropriate address components by the locator.

```
var matches = await _geocodeTask.GeocodeAsync("380 New York Street, Redlands CA 92373");
```

Copyright © 1995-2020 Esri. All rights reserved.

You can also geocode using a result from an address suggestion. The following example geocodes a suggested address as created in the Get address suggestions section of this topic.

```
// get the selected suggestion in a ListBox
var suggestion = SuggestionsListBox.SelectedItem as SuggestResult;
var matches = await _geocodeTask.GeocodeAsync(suggestion);
```

The following example matches a Japanese address with the ArcGIS World Geocoding Service. The geocode parameters restrict the address search to Japan.

```
// set geocode parameters to search within Japan
var geocodeParams = new GeocodeParameters
{
    CountryCode = "Japan",
    OutputLanguage = new System.Globalization.CultureInfo("ja-JP")
};

geocodeParams.ResultAttributeNames.Add("City");

// geocode the Japanese Prime Minister's office address
var matches = await _geocodeTask.GeocodeAsync("東京都千代田区永田町２-２-２", geocodeParams);

// get the top match (first one), return if none are found
// (add a using statement for System.Linq to enable syntax like the following)
var bestMatch = matches.FirstOrDefault();
if (bestMatch == null) { return; }
```

Note:
See Geocode coverage in the REST API documentation for a list of supported countries and corresponding codes.

Find points of interest

You can also geocode points of interest using a place name, such as `Eiffel Tower` or even more general terms, such as `bank` or `bus stop`. The geocode parameters allow you to restrict your search to a specified extent or to prefer candidates that are closer to a specified location, which can be useful if searching for facilities near the user's current position. You can provide the user with information about the locations, such as addresses, phone numbers, and URLs for candidates, as well as the distance from a reference location.

Copyright © 1995-2020 Esri. All rights reserved.

Tip:

Not all locators support geocoding a point of interest. The result information for candidates will also vary with the locator. Use `LocatorInfo` to see if geocoding points of interest is supported and which attributes are available for the result candidates, as described in a preceding section. The `Distance` attribute will only contain (non-zero) values if a preferred search location (point) is provided in the geocode parameters.

Geocoding a point of interest involves the same process as using an address described in the previous section. Instead of passing a single-line address value as input, you can provide a place name. The following are examples:

- Point of interest by name
  - `Wrigley Field`
  - `Disneyland`
  - `mount everest`
- Point of interest by category
  - `restaurant`
  - `petrol`
  - `coffee in Salt Lake City`
- Administrative place name, such as a city/county/state/province/country name—`Seattle, Washington`
- Postal code—`92591 USA`

You can use geocode parameters to refine or restrict search results when geocoding place names just as you do with addresses.

The following example searches for places in the category `Ice Cream Shop` near a specified location (the center of the map extent) and gets the name, phone number, address, and distance of the closest match.

```
// return if points of interest are not supported by this locator
if (!_geocodeTask.LocatorInfo.SupportsPoi) { return; }

// set geocode parameters to return a max of 5 candidates near the center of the current extent
var mapCenter = MyMapView.GetCurrentViewpoint(ViewpointType.CenterAndScale).TargetGeometry as
MapPoint;
var geocodeParams = new GeocodeParameters
{
    MaxResults = 5,
    PreferredSearchLocation = mapCenter
};

// request address, phone, and distance as result attributes
// (note: distance is only populated if a PreferredSearchLocation is set in the geocode
parameters)
geocodeParams.ResultAttributeNames.Add("Place_addr");
geocodeParams.ResultAttributeNames.Add("Phone");
geocodeParams.ResultAttributeNames.Add("Distance");

// find candidates using a place category ("ice cream")
var matches = await _geocodeTask.GeocodeAsync("ice cream", geocodeParams);

if (matches.Count == 0) { return; }

// get information for the closest candidate
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// (add a using statement for System.Linq to enable syntax like the following)
var closestMatch = (from m in matches orderby m.Attributes["Distance"] select
m).FirstOrDefault();
var name = closestMatch.Label;
var address = closestMatch.Attributes["Place_addr"].ToString();
var phone = closestMatch.Attributes["Phone"].ToString();
var distance = (double)closestMatch.Attributes["Distance"];
```

Related topics

Find a route

Copyright © 1995-2020 Esri. All rights reserved.

Search for related features

You can search for features in one table that are related to features in another table, provided a relationship was defined between the two tables. Table relationships are defined using ArcGIS Desktop, as described in Essentials of relating tables in the ArcGIS Desktop help.

Note:
For example, say your app lets users view all the fire hydrants (fire hydrant features) in a city through a **hydrants** table. You now want users to be able to look up the inspection reports for a given hydrant that are stored in a different table, say **hydrant-inspections**. If a relationship has been defined between these tables, you can query the **hydrants** table to find related features from the **hydrant-inspections** table for a given hydrant.

The following steps describe how to set up the above scenario — searching for inspection reports (related features) of a given hydrant (feature). Note that you could also query the relationship in the other direction to search for all fire hydrants with a specified inspection criteria.

Search for related features

1. Add the related tables to the same map. If the tables participating in the relationship are in a single map service, you can add the service to the map as an `ArcGISMapImageLayer`. This will add all spatial and non-spatial tables from the service. If a table is a spatial table from a map service or feature service, you can add it to the map's operational layers as a feature layer. Likewise, a non-spatial table can be added to the map's table collection. In a disconnected scenario, you add the tables to the map after downloading them into a mobile geodatabase.

   ```
   FeatureLayer hydrantsLayer = new FeatureLayer(hydrantsTable);
   map.OperationalLayers.Add(hydrantsLayer);
   map.Tables.Add(inspectionsTable);
   ```

   All participating tables must be in the same map but they do not have to be loaded..

2. If you want to return only those features from the table that participate in the provided relationship, set input parameters as described in this step. Otherwise, skip to step 4.
   a. Create a `RelatedQueryParameters` object.
   b. Provide a value for the `RelationshipInfo` property. Any time you create a `RelatedQueryParameters` object, you must also provide a `RelationshipInfo` that defines the relationship.

   ```
   // Get the first relationship info for the hydrants service feature table (inspections)
   RelationshipInfo relateInfo = hydrantsTable.LayerInfo.RelationshipInfos.FirstOrDefault();

   // Create a new RelatedQueryParameters to find related features with a status of 'Failed'
   RelatedQueryParameters queryParams = new RelatedQueryParameters(relateInfo);
   queryParams.WhereClause = "InspectionStatus = 'Failed'";
   ```

   c. You can provide additional input parameters, such as a `WhereClause`, to further filter the related features, whether or not you wish to receive the geometry of the related features, and result sorting preferences.

3. Query the table containing the feature for which you want related features. In the hydrant example, you would query the **hydrants** table to find inspection reports for a particular hydrant.

Copyright © 1995-2020 Esri. All rights reserved.

```
IReadOnlyList<RelatedFeatureQueryResult> relatedQueryResults = await
hydrantsTable.QueryRelatedFeaturesAsync(selectedHydrant, queryParams);
foreach (var result in relatedQueryResults.ToArray())
{
    foreach (var inspection in result.ToArray())
    {
        // Load the related feature to bring in all field values
        await (inspection as ArcGISFeature).LoadAsync();

        // .. Process each inspection record for this hydrant ..
    }
}
```

In a connected scenario, whether a query is online or local depends on the feature request mode of the related table. For more information about feature request modes, see the Table performance section in the Layers and tables topic.

- For `OnInteractionCache`, the query can be online or local as features are cached
- For `OnInteractionNoCache`, the query is always online
- For `ManualCache`, the query is always local

In a disconnected scenario, all queries are local.

4. If you want to find related features from **all** participating relationships of a given feature, perform the search without any query parameters.

```
// Call the QueryRelatedFeaturesAsync overload that doesn't take a RelatedQueryParameters
IReadOnlyList<RelatedFeatureQueryResult> relatedQueryResults = await
hydrantsTable.QueryRelatedFeaturesAsync(selectedHydrant);
foreach (var result in relatedQueryResults.ToArray())
{
    foreach (var inspection in result.ToArray())
    {
        // Load the related feature to bring in all field values
        await (inspection as ArcGISFeature).LoadAsync();

        // .. Process each inspection record for this hydrant ..
    }
}
```

5. Handle query results.
   Query operations are asynchronous and get called back with the query results and an optional error, if any. The results are an array of `RelatedFeatureQueryResult` objects, one for each related table that was queried. If more than one table was queried, multiple query operations may be performed internally and the query method is called back when all such operations have completed.

   Each `RelatedFeatureQueryResult` contains a list of related features that can be iterated over. The query result also contains the `RelatedTable` and `RelationshipInfo` since multiple instances of the same table can be added to the map (for instance with different scale level visibility, definition expression, and so on). In such cases, a query is performed on each table instance and a `RelatedFeatureQueryResult` object is returned for each.

   The query result also contains:
   - The feature for which the related features were queried.
   - Information on whether the resulting related feature count exceeded the maximum limit supported by the server.

Copyright © 1995-2020 Esri. All rights reserved.

Edit features

Copyright © 1995-2020 Esri. All rights reserved.

Editing

Editing means that you can add, update, and delete features. For more information on features see Features and graphics. Updating features includes modifying feature attributes (changing the value of a feature's fields according to the type and range of values allowed); modifying feature geometry (such as moving a point feature or reshaping a polygon or polyline feature); and if available adding, updating, and deleting feature attachments (related files such as pictures, documents or videos). Feature layers inside your map's operational layers collection provide the basis for editing. Feature layers store and edit their data using database tables called feature tables.

Editing features is supported in a number of workflows, described below:

- Online feature service editing, where a table is created from the service, edits are made, and the changes are applied back to the service as soon as the user has finished the edit operation. Suitable for multi-user editing scenarios.
- Offline feature service editing and sync, where a local geodatabase is created from a sync-enabled feature service before going offline, tables are retrieved from the geodatabase while offline, edits are made, then changes are applied back to the service when the device is back online again (and server changes are optionally synchronized back). Suitable for multi-user offline workflows. See Work with offline maps for more information on offline workflows.
- Static feature collection editing, where tables are created from the features in the map or a portal item, edits are made and changes are saved back into the map or portal item. This is a suitable workflow for sharing static data to lots of clients, but should not be used to edit data across a number of clients or to share data that changes frequently.
- Offline file-based editing. Edits can be made to features from a GeoPackage file (.gpkg) that supports the OGC GeoPackage specification. Such a file is stand-alone and is not backed by a feature service; it is fully offline. If your workflow includes sharing edits online and managing edits across many users, it is recommended that you use online feature service editing.

See Perform edits for more information.

Note:
ArcGIS Runtime also supports editing KML. This workflow differs from those described here for editing features in a feature layer. See the topic Edit KML content for details.

Creating feature services for your data

Feature services provide the symbology, feature geometry, and set of attribute fields (schema) for your features. Feature services contain service layers (feature data) and tables (nonspatial, tabular data) that you edit via a feature table for both online and offline feature service workflows. Feature services allow for scalable multi-client editing for data which changes over time. For more information about feature services, see What is a feature service?

Before you build an app that performs feature service editing, you need a feature service that exposes the layers you want to edit. There are various ways to publish a feature service.

- You can login to your organization's portal and publish features from a variety of data sources such as CSV files, GeoJSON, shapefiles, feature collections, or file geodatabases. All of these options and their steps are outlined in the publish hosted feature services topic.
- You can publish a feature service using ArcGIS Desktop. This involves setting up a map document and defining a feature class schema, or importing an existing feature class. Optionally, you can define feature templates to make it easy for the app's end user to create common features. You can then publish the map as a service to ArcGIS Online or ArcGIS Enterprise. For offline workflows, you must enable the **Feature Access** capability and ensure that the service is sync-enabled (discussed in Prepare data for offline use.

Copyright © 1995-2020 Esri. All rights reserved.

- You can create a feature layer using the ArcGIS for Developers site. Log in to the site using your organization account or your free developers subscription. Access the **Layers** tab and click the **Create New Layer** button. For offline workflows, follow the instructions to create the new feature layer ensuring that you have checked the box to enable the layer to be taken offline to allow it to be viewed, edited, and synchronized. After creating a layer you can add data to it in the ArcGIS Online Map Viewer.

However you publish your service, REST URLs (endpoints) are created to both a map service and a feature service. Use these URLs to reference the services in your app. The map service endpoint can be used for read-only access to features (viewing only), so make sure to reference the feature service endpoint for editing. Offline editing workflows require sync-enabled feature services. A sync-enabled feature service allows you to generate a geodatabase for offline use, and gives you the ability to sync your local edits back to the service.

Note:
ArcGIS 10.2.2 for Server or later, or hosted services in ArcGIS Online, are required to publish sync-enabled feature services.

Feature collections

Feature collections are static collections of features stored as JSON inside the map or a referenced portal item. Many ArcGIS Online operations create feature collections, such as adding Map Notes to your map, importing shapefiles, importing GPX files or sharing analysis results as items. There are two representations of feature collections which are important to understand when it comes to editing and saving features.

1. Feature collections in a map - this is where the feature JSON is saved in a single map as a feature collection layer.
2. Feature collections as portal items - this is where the feature JSON is saved as a portal item. These feature collections can be referenced in multiple maps.

The workflow for editing the features for both types of feature collections is the same. However, there are differences when persisting the edits so that other users can see them. Feature collections in a map will be persisted when the map is saved, others who open the map will see the edited features. Feature collections stored as portal items will not be saved when a map is saved, you will have to update or save a portal item to ensure that others will see the edited features.

Feature collections should not be used for multi-user editing scenarios, as clients could easily hold on to older versions of the feature collection and overwrite each others changes. Use feature services for these types of workflows.

Perform edits

Fine grained control over editing operations is available by using the editing API, allowing you to create and edit features, add, edit or remove feature attachments, and edit the geometry of features. For editing workflows that use a local geodatabase, you can use geodatabase transactions to manage a set of edits (transaction). You can then control when those edits are **committed** (saved) or **rolled back** (discarded) as a single unit.

For some feature service editing workflows, it's a good idea to have an analyst using ArcGIS Desktop periodically review the edits to verify data integrity. Although components in the API can perform some data validation, other tasks such as validating topologies cannot be performed using ArcGIS Runtime SDK alone.

Editor tracking

The editing framework supports the tracking of specific edits to features. This happens by tracking the following feature properties:

- Which user created the feature
- Date and time the feature was created

Copyright © 1995-2020 Esri. All rights reserved.

- Which user last edited the feature
- Date and time the feature was last edited

Related topics

Edit features
Edit geometries
Sync offline edits

Copyright © 1995-2020 Esri. All rights reserved.

Edit features

ArcGIS Runtime supports editing workflows that leverage the use of features. A feature is a persisted piece of data (essentially a row in a table) that includes a geographic location (shape) as well as attributes that further describe it, such as Name, Area, Population, ZoningCode, and so on. Your app can include the ability to add and delete entire features or to make edits to existing feature geometry and attributes. If your features have attachments (images or text files, for example) or related records in another dataset, that information can be edited as well.

Features can be hosted in an online service, stored locally in a database, or saved in a map or portal item. How your features are stored affects how you make and manage edits to them in your app. For more information on features and where they come from, see Features and graphics.

Editing overview

Your apps can edit features while online (connected to the network and using data hosted online) or offline (without a network connection and using local data). You can edit features from a service, a geodatabase, or a feature collection. For an introduction see Editing. While all editing workflows are described in this topic, note that there is no difference in the code that actually makes the edits (adding, deleting, or updating features, in other words).

Features from an `ArcGISFeatureTable` return `ArcGISFeature` objects which implement the loadable pattern for increased efficiency. When fetching `ArcGISFeature` objects for rendering and query purposes, a minimum set of required fields is returned, such as identifiers, geometry, fields used for symbolizing, and so on. When you want to edit the feature, you must load it first, otherwise the edit operation will fail.

Note:
When querying features from an online service, you have the option to return your results as loaded features so all fields are immediately available and edits can be performed.

Feature editing is supported for a number of workflows. The developer patterns for each is described below:

Tip:
For editing workflows that use a local geodatabase, you can use geodatabase transactions to manage a set of edits (transaction). You can then control when those edits are **committed** (saved) or **rolled back** (discarded) as a single unit.

The online feature service editing workflow is:

1. Create a service feature table from a feature service using the REST endpoint (URL, in other words) to identify the features you want to work with, or open a map which already contains a service feature table via a feature layer.
2. Perform any edits against the service feature table (add, update, and delete features and feature attachments).
3. Apply your edits to the feature service right away.

The offline feature service editing workflow is:

1. Generate a local runtime geodatabase using a sync-enabled feature service while the user is online as described in Take a layer offline.
2. Create a geodatabase feature table from the local geodatabase. If this is for a connected feature layer already in a map, then that layer will need to be switched out with a new feature layer created with this table.
3. Perform any edits against the geodatabase feature table (add, update, and delete features and feature attachments).

Copyright © 1995-2020 Esri. All rights reserved.

4. Synchronize your edits with the service once you have a network connection, as described in Sync offline edits.

The static feature collection editing workflow is:

1. Create a new feature collection or open a map which already contains a feature collection via a feature collection layer.
2. For new feature collections, create a new feature collection table and add it to the feature collection. For feature collections from a map, access a feature collection table from the feature collection.
3. Perform any edits against the feature collection table (add, update, and delete features).
4. For feature collections which are to be stored in the map, save the map. For feature collections to be saved as a portal item, save the feature collection as a portal item.

Create a feature table

Before you can edit features, you need to obtain feature data from a feature table.

In a connected workflow, the type of feature table you use and edit is a `ServiceFeatureTable`. In a disconnected workflow, the type of feature table you use and edit is a `GeodatabaseFeatureTable`. Both classes inherit from `ArcGISFeatureTable`, which itself inherits from `FeatureTable`.

The class hierarchy between these is as follows:

For working with features by value, the type of table you use is `FeatureCollectionTable`. Feature collection tables do not contain attachments and inherit directly from `FeatureTable`.

The methods to add, update, and delete features are common to `FeatureTable` and the methods for updating attachments are provided by `ArcGISFeatureTable`. Therefore, all the methods you have at your disposal to edit features from a feature service are available both online and offline. The `ServiceFeatureTable` used in connected editing provides additional methods to apply feature edits and attachment edits back to the service. You should apply any edits to the service as soon as you have made them on your feature table.

Feature service edit capabilities

When working with features from a feature service online or offline, additional edit capabilities are defined to allow or prevent certain types of edits to features. You should check these capabilities to determine what editing capability is available. You may also need to verify ownership-based access is available for each edit you want to complete.

Copyright © 1995-2020 Esri. All rights reserved.

You can use properties on a feature table to find out what type of editing is supported.

- Is editable: indicates if the table is editable.
- Edit capabilities: describes the types of edits that are allowed for the table, such as adding, deleting, or updating features, for example.

A feature service might also use ownership-based access control, which means edits to particular features are only available to certain users. You can check ownership-based access using properties on the feature table.

- Can add: whether or not the current user can add a feature to the table.
- Can delete: whether or not the current user can delete a feature from the table.
- Can update: whether or not the current user can edit an existing feature.

Online feature services

In an online workflow, you need a service feature table (`ServiceFeatureTable` class) created from a feature service. If you have opened a map, you can use a feature layer and retrieve its feature table and check it is of the correct type. For new layers, you can create a service feature table with the feature service URL and then create a feature layer. Once the feature layer is in a map and displayed in a map view, features in the map extent are added to the service feature table and can be edited. As you pan and zoom the map (that is, change the map extent), more features are retrieved from the feature service and added to the service feature table. If you are not using a map or you want to pre-populate your table with a fixed set of features, use the `populateFromService` method.

Note:
The number of features you see when you initially visit a new area of the map is limited by the feature service's maximum record count property (the default value for this property is 1,000). Pan and zoom to query the service for more features. Once features are retrieved from the service and visible in your map, they are available in your service feature table for editing and querying.

Use code like the following code to instantiate a new service feature table from a URL.

```
// create a new geodatabase feature service table
var featureTableUri = new System.Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/
DamageAssessment/FeatureServer/0");
var table = new ServiceFeatureTable(featureTableUri);
```

Because they rely on a remote service to properly initialize their state, service feature tables use the Loadable pattern. You can load a service feature table and check its load status before adding it to the map, which allows you to handle load failure and

Copyright © 1995-2020 Esri. All rights reserved.

to retry loading the table if necessary. However, if you use the feature table to create a feature layer and add that to a map or scene, the table will automatically attempt to load when the layer is displayed.

Note:
Features in a service feature table are Loadable for added efficiency; before editing your feature ensure it is loaded otherwise the edit operation will fail.

Offline feature services

In an offline workflow, you need to first generate a runtime geodatabase from a sync-enabled feature service while you have a network connection. Follow the geodatabase generation process described in Take a layer offline. This process results in a geodatabase stored locally on disk. The geodatabase contains one or more feature tables, one for each service layer or service table that you requested in the geodatabase generation process. When offline, create geodatabase feature tables (`GeodatabaseFeatureTable` class) from the geodatabase, for example, on application load. The features stored in the geodatabase tables are available for editing whether you display them in a layer or not, but in most cases you'll want to create feature layers to display them for editing. Also, consider generating an offline basemap to give the feature data some geographical context when your users edit offline.

Caution:
You can also generate a runtime geodatabase using ArcGIS for Desktop, but these geodatabases are read-only.

The following example shows several ways to get a feature table from a local geodatabase:

```
// open a local geodatabase
var gdb = await Esri.ArcGISRuntime.Data.Geodatabase.OpenAsync(gdbPath);

// get the first table in the database
var table = gdb.GeodatabaseFeatureTables.FirstOrDefault();

// get the third table
var anotherTable = gdb.GeodatabaseFeatureTables.ElementAt(2);

// find a table by name
var marineTable = gdb.GeodatabaseFeatureTables.FirstOrDefault(t => t.TableName == "Marine");

// loop through all tables
foreach (var t in gdb.GeodatabaseFeatureTables)
{
    // ... code here to work with the table ...
}
```

Static feature collections

For working with static feature collections, you either need a map which already contains a feature collection layer, or you need to create a new feature collection layer.

To create a new feature collection layer with a feature collection:

```
// Open a portal item containing a feature collection
var portal = await ArcGISPortal.CreateAsync();
var collectionItem = await PortalItem.CreateAsync(portal, itemId);

// Verify that the item is a feature collection
```

Copyright © 1995-2020 Esri. All rights reserved.

```
if (collectionItem.Type == PortalItemType.FeatureCollection)
{
    // Create a new FeatureCollection from the item
    var featureCollection = new FeatureCollection(collectionItem);
          // Create a layer to display the collection and add it to the map as an operational
layer
    var featureCollectionLayer = new FeatureCollectionLayer(featureCollection);
    featureCollectionLayer.Name = collectionItem.Title;
    MyMapView.Map.OperationalLayers.Add(featureCollectionLayer);
}
```

Add layers to the map

You can display the features contained in the feature table by adding them to a feature layer in the map. Create feature layers from any feature table then add them to a map. A feature layer, once added to a map, takes care of displaying the features contained in your feature table within the map extent currently displayed. When features in the table are edited, the edits are visible right away in the associated feature layer, but not immediately committed back to the feature source.

The following example calls `LoadAsync`, then checks the `LoadStatus` to verify the table was loaded successfully from the service or geodatabase. It then creates a layer from the feature table and adds it to the map.

```
// load the feature table
await table.LoadAsync();

// if the table was loaded successfully, create a new feature layer for the table and add it to
the map
if (table.LoadStatus == Esri.ArcGISRuntime.LoadStatus.Loaded)
{
    var lyr = new FeatureLayer(table);
    MyMapView.Map.OperationalLayers.Add(lyr);
}
```

You will be able to see features like this on your map.

Add features

For creating new features, it's common for an app to allow the user to click the map to specify a new feature's location. You can provide this capability by listening to a click event on your map view, which in turn will call a function for adding a new feature.

```
private void MyMapView_GeoViewTapped(object sender,
Esri.ArcGISRuntime.UI.Controls.GeoViewInputEventArgs e)
{
    // get the click point in geographic coordinates
    var mapClickPoint = e.Location;
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    // call a function that will add a new feature at this location
    AddDamageFeature(mapClickPoint, "Minor", "Earthquake");
}
```

To add features to a feature table, create a new feature from geometry (for example, point, line, or polygon), create attributes for the new feature, and then call add feature. This adds the feature to a table stored locally on your device. Even if you're editing a service feature table, your edits are initially stored in a table on the client. You must explicitly apply service feature table edits to commit them to the parent feature service.

Note:
All features you add or update in a feature service should have valid geometry. Creating features with empty (null) geometry can cause unexpected issues when working with features in ArcGIS Runtime offline editing workflows.

```
private async void AddDamageFeature(MapPoint structureLocation, string type, string cause)
{
    var attributes = new Dictionary<string, object>();
    attributes.Add("typdamage", type);
    attributes.Add("primcause", cause);

    // create a new feature in the damage table, pass in the attributes and geometry
    var newFeature = _damagePointsTable.CreateFeature(attributes, structureLocation);
    // add the new feature (this updates the local copy of the table)
    await _damagePointsTable.AddFeatureAsync(newFeature);

    // push this update (apply edits) to the feature service
    IReadOnlyList<EditResult> editResults = await _damagePointsTable.ApplyEditsAsync();
    // check the results for errors
    foreach (var r in editResults)
    {
        if (r.CompletedWithErrors)
        {
            Console.WriteLine("Edit to Object '" + r.ObjectId + "' failed: " + r.Error.Message);
        }
    }
}
```

Tip:
When adding new features, it is good practice to ensure the geometry of the new feature is not null. If the location of a feature is not yet known, you may want to use an empty geometry.

Select features

In an editing app, a common workflow is for the user to choose features to edit by selecting them with a query or interactively on the map. Features in a feature layer can be selected, which results in them being highlighted in the map view. You can select features based on spatial relationships (sample points located inside the forest boundary polygon), using attribute criteria (pipes with a material_type attribute of 'pvc'), or both (parcels outside the city limits with a zoning code of 'COM'). You can also get selected features from a layer, after letting the user make an interactive or custom selection, for example.

Copyright © 1995-2020 Esri. All rights reserved.

In editing workflows, features are typically selected for tasks such as deleting or updating.

```
// Build a buffer (polygon) around a click point
var buffer = GeometryEngine.Buffer(queryPoint, 5000);

// Use the buffer to define the geometry for a query
var query = new QueryParameters();
query.Geometry = buffer;
query.SpatialRelationship = SpatialRelationship.Contains;

// Select features in a feature layer using the query
await damageLayer.SelectFeaturesAsync(query, Esri.ArcGISRuntime.Mapping.SelectionMode.New);
```

You can also select specific features using the select feature method and passing in a single feature.

Update features

Feature updates include moving or reshaping a feature's geometry or making edits to attribute values. As with all editing operations the changes are not automatically committed to the features source, it is the developer's responsibility to do this.

```
// get all selected features in the layer
var damageLayer = MyMapView.Map.OperationalLayers[0] as FeatureLayer;
var selectedFeatures = await damageLayer.GetSelectedFeaturesAsync();

// loop thru all features in the selection
foreach (ArcGISFeature f in selectedFeatures)
{
    // load the feature
    await f.LoadAsync();

    // update the damage type attribute
    f.Attributes["typdamage"] = "Inaccessible";

    // get the feature's geometry (point), move it to the north
    var location = f.Geometry as MapPoint;
    var newLocation = new MapPoint(location.X, location.Y + 50000);

    // set the feature's new location
    f.Geometry = newLocation;

    // update the feature in the local cache with these edits
    await _damagePointsTable.UpdateFeatureAsync(f);
}
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// apply all the edits back to the service feature table
var editResults = await _damagePointsTable.ApplyEditsAsync();
```

Delete features

You can delete several features from a feature table using the delete features method that accepts a list of features, or just a single feature with a call to delete feature. As with all editing operations the changes are not automatically committed to the features source, it is the developer's responsibility to do this.

```
// get all selected features in the layer
var damageLayer = MyMapView.Map.OperationalLayers[0] as FeatureLayer;
var selectedFeatures = await damageLayer.GetSelectedFeaturesAsync();

// delete the selected features from the local cache
await _damagePointsTable.DeleteFeaturesAsync(selectedFeatures);

// apply all the edits back to the service feature table
var editResults = await _damagePointsTable.ApplyEditsAsync();
```

Feature attachments

Attachments are files you associate with a feature, including text, images, videos, and so on. Attachments are only available for features from an ArcGIS feature table. You might want to attach a photograph to a parcel feature, for example, to show a picture of the property. Attachments on a feature are manipulated using the following methods on the ArcGIS feature class.

Method	Description
add attachment	Adds a new attachment to a feature
delete attachment	Deletes a single existing attachment
delete attachments	Deletes a list of existing attachments
update attachment	Updates an existing attachment
fetch attachments	Accesses the attachments for a feature

ArcGIS feature inherits from the feature class, so you can cast from feature to ArcGIS feature if needed. Prior to adding, updating or deleting features it is good practice to make sure that you have permission to perform one of these operations. This is done by using the can edit attachments property. Once you have performed your attachment edits, these are applied to the feature using the update feature method and finally the feature needs to be applied to the service by calling the apply edits method on your feature table.

```
// cast to ArcGISFeature to add attachments
ArcGISFeature agsFeature = (ArcGISFeature)feature;

// load the feature
await agsFeature.LoadAsync();

// add an attachment - pass the name, mime type, and attachment data (bytes, e.g.)
await agsFeature.AddAttachmentAsync("AssessmentImage.jpg", "image/jpg", imageData);

// save the attachment with the feature
await _damagePointsTable.UpdateFeatureAsync(agsFeature);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
//commit update operation
IReadOnlyList<EditResult> results = await _damagePointsTable.ApplyEditsAsync();

// check edits for errors
var errs = from e in results where e.CompletedWithErrors select e;
foreach (var err in errs)
{
    // process errors here
}
```

Related topics

Editing
Edit geometries
Sync offline edits

Copyright © 1995-2020 Esri. All rights reserved.

Edit geometries

You edit geometries when you want to change existing geometries or create new ones. However, `Geometry` objects are immutable. To edit geometries, use geometry builders.

Geometry builders

Geometry builders (or builders for short) create or change geometry. A geometry builder contains the same things a geometry contains—vertices, segments, and parts—allowing its state to be changed as needed. You can create new geometries at any point from the builder's current state. A builder is available for each type of geometry. Each builder exposes the appropriate methods for modifying a specific type of geometry. In the case of `PolylineBuilder` and `PolygonBuilder` that share many members, both inherit from `MultipartBuilder`, which in turn inherits from `GeometryBuilder`. Other builders inherit directly from `GeometryBuilder`.

Because builders represent geometries under construction, they do not use the same immutable collection types as those returned from members of the geometry classes containing a part, part collection, or point collection. Instead, builder members use mutable collection types.

Modify existing geometry

To modify an existing geometry, complete the following steps:

1. Create the appropriate type of builder and initialize its state by passing in the geometry to be edited to the constructor. For example, modify a polyline by creating an `PolylineBuilder` and pass in the existing polyline to the builder.
2. Use the methods on the builder to change the builder's state. For example, add a new part to a polyline or remove a point from a multipoint.
3. Call `ToGeometry` to return a new immutable geometry from the current state of the builder.
4. Repeat steps 2 and 3 as many times as necessary. By keeping a reference to the builder, you can repeatedly modify the builder state as many times as required, for example, to update a temporary graphic to display the current state to the user.

The following code demonstrates an `PolylineBuilder` held as a member variable in a custom geometry editor class. The `AddPointToEnd` method adds another vertex to the end of the polyline and can be called when a user taps at the required location. Each time an update is made, the `UpdateRoadGraphic` method can be used to update an `GraphicsOverlay` with the new state of the geometry created so far.

```
public class RoadGeometryEditor
{
    PolylineBuilder polylineBuilder;
    Graphic roadGraphic;

    public RoadGeometryEditor(GraphicsOverlay roadsOverlay, Feature roadFeature)
    {
        // create a PolylineBuilder for working with road geometry
        // set initial state of the builder based on the Polyline passed in
        var roadPolyline = roadFeature.Geometry as Polyline;
        this.polylineBuilder = new PolylineBuilder(roadPolyline);

        // create a graphic to show the road geometry
        var lineSymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Solid,
Color.FromArgb(0,255,0,0), 1.0);
        this.roadGraphic = new Graphic(roadPolyline, lineSymbol);

        // display the graphic in a graphics overlay in the map view
```

Copyright © 1995-2020 Esri. All rights reserved.

```
        roadsOverlay.Graphics.Add(roadGraphic);
    }

    public void AddPointToEnd(MapPoint point)
    {
        // add a point to the end of the last part in the polyline
        this.polylineBuilder.AddPoint(point);
    }

    // a read-only property to get the current Polyline stored in the builder
    public Polyline RoadGeometry
    {
        get { return this.polylineBuilder.ToGeometry(); }
    }

    // update the line graphic with the geometry currently stored in the polyline builder
    public void UpdateRoadGraphic()
    {
        this.roadGraphic.Geometry = this.polylineBuilder.ToGeometry();
    }
}
```

You don't have to initialize a builder with an existing geometry—you can also use builders to create entirely new geometries.

When creating geometries, use IsValidSketch to determine when a builder contains enough information to convert the builder's state successfully to a geometry.

Tip:

When editing geometries using locations from the screen, and the map is a wraparound map, you may need to normalize the geometry before you save it to a geodatabase, or use it in tasks or other operations.

Spatial reference management

When working with geometries and builders, the following principles apply with regard to their `SpatialReference`:

- The spatial reference of a builder is always set during builder instantiation and cannot be changed. It can be set to null.
- You don't need to set the spatial reference of points or segments passed into methods on geometries, collections, or builders. If the spatial reference is not set, the input coordinates will be assumed to be in the same spatial reference as the geometry, collection, or builder.
- It is an error to add a point or segment having a different spatial reference than that of the geometry, collection, or builder you're adding it to, and an exception will be thrown indicating a mismatch. This includes the case where the spatial reference of the geometry, collection, or builder is null and that of the added point or segment is not null.
- A `MapPoint` or `Segment` returned from a geometry's member will have the same spatial reference as the geometry it came from.
- When creating a new `Polygon` using the constructor, the API first sets the spatial reference parameter for the new `Polygon`, then the points are added. If there is a mismatch between the spatial reference of the new `Polygon` and the points being added, this is an error and the constructor will throw an exception.

Tip:

If required for your workflow, use the method on `GeometryEngine` to project points to a different spatial reference.

Copyright © 1995-2020 Esri. All rights reserved.

Multipoint geometries

Use `MultipointBuilder` to modify an existing `Multipoint`. Get and modify the mutable point collection from the builder's `Points` property, working directly with this collection to add, remove, or insert points. Call `ToGeometry` to instantiate a new immutable instance of `Multipoint` from the current state of the builder.

```
var builder = new MultipointBuilder(SpatialReference.Create(2169));
builder.Points.Add(pointOne);
builder.Points.Add(pointTwo);
builder.Points.Add(pointThree);
// ...
var myMultipoint = builder.ToGeometry();
```

Multipart geometries

Polygons and polylines can have multiple parts. You can use a multipart polygon to define a group of islands, or a multipart polyline to represent a non-continuous linear feature. Due to the geometric complexity of polygons and polylines, `PolylineBuilder` and `PolygonBuilder` are used more frequently than other builders such as `PointBuilder`. Using the methods on these collections, you can build polygons and polylines by adding points. As with multipoint geometries, use the `PointCollection` to modify multipart geometries. Or, you can edit the segments in multipart geometries.

```
// create a new builder for creating a polygon in geographic coordinates
var builder = new PolygonBuilder(SpatialReferences.Wgs84);

// create a part for the polygon
var partOne = new Part(builder.SpatialReference);

// add a line segment going from (0,0) to (0,7)
partOne.Add(new LineSegment(0, 0, 0, 7));
// add another segment from (0,7) to (3,6)
partOne.Add(new LineSegment(new MapPoint(0, 7), new MapPoint(3, 6)));

// add this part to the polygon builder
builder.AddPart(partOne);

// get the polygon from the builder: the segments will be closed to create a triangle
var myPolygon = builder.ToGeometry();
```

Copyright © 1995-2020 Esri. All rights reserved.

You can also edit using a mix of point-based and segment-based methods.

```
// add a new part to the polygon builder
partOne = new Part(builder.SpatialReference);
builder.Parts.Add(partOne);

// add an initial point
partOne.AddPoint(0, 0);
// add another point to create a segment from (0,0) to (0,7)
partOne.AddPoint(0, 7);
// add a line segment from (0,7) to (5,0)
partOne.Add(new LineSegment(0, 7, 5, 0));
```

Editing parts

Use the `PartCollection` methods to add, remove, or insert `Parts` to a `MultipartBuilder`. A point-based helper method `AddPart(IEnumerable<MapPoint>)` is available for the common task of adding an existing set of `MapPoints` to a `PartCollection` as a new `Part`.

```
var builder = new PolygonBuilder(SpatialReferences.Wgs84);
// create a list of points to define the first part (ring) for the polygon
var partOnePoints = new List<MapPoint>();
partOnePoints.Add(new MapPoint(-112.33, 38.51));
partOnePoints.Add(new MapPoint(-112.53, 38.35));
partOnePoints.Add(new MapPoint(-112.73, 38.18));
// add the points to the builder as a part
builder.AddPart(partOnePoints);
```

Editing parts using points

The `Part` is a collection of segments. Working with the point-based method results in defining an ordered series of vertices and, as a consequence, creating straight `LineSegments` within the builder between those vertices.

The following principles apply when editing parts using points:

- Use `InsertPoint(int, Point)` to insert a vertex at a specific index into a part. This will increase the number of segments in the part by one. Connections between segments are preserved—adjacent segments touch at their starting and ending vertices.

  ```
  // insert a new vertex into the first part (ring) before the vertex at index position 2 (3rd
  one)
  builder.Parts[0].InsertPoint(2, new MapPoint(-112.60, 38.22));
  ```

- Use `RemovePoint(int)` to remove the vertex at a specific index from a part. The number of segments decreases by one, and connections between segments are preserved.

  ```
  // remove the first vertex (index position 0) from the first part (ring)
  builder.Parts[0].RemovePoint(0);
  ```

- Use `SetPoint(int)` to replace a vertex in the collection without the need for a remove followed by an add. The total number of vertices or segments in the collection remains unchanged.

Copyright © 1995-2020 Esri. All rights reserved.

- You can add a new vertex to the end of a `MultipartBuilder` using the `AddPoint(int, Point)` method. This adds a vertex to the end of the last part in the builder. If there are no parts in the builder, this method will create a new part first, then add the new vertex.
- You do not need to close a part belonging to a polygon.

⚠ Caution:

When working with point-based methods that take an index parameter, keep in mind that this is the index of the specific vertex, which is a different index than that used when working with segments. This is because where a segment's ending vertex is at the same location as the starting vertex of the next segment, it is represented by a single `MapPoint` in point-based methods.

Editing parts using segments

When you work with the segments in a part, you specify each segment's starting and ending vertex. This differs from editing with points, where sequential vertices define a series of adjacent segments. Specifying starting and ending vertices for adjacent segments can result in gaps, where the end vertex of one segment does not have exactly the same location as the start vertex of the next adjacent segment. Gaps are not automatically corrected in the builder until you access the builder's call `ToGeometry` member to instantiate a new polyline or polygon. At that time, if any adjacent segments in the part do not touch, a short connecting segment will be added automatically to the new geometry to ensure the part forms a continuous edge in the new geometry. To avoid automatically adding segments, re-use the location of the end vertex of the previous segment as the start vertex of the subsequent segment, and use insert, remove, and add methods with care.

Similar to geometries, all segments are immutable and are created by using constructors or static factory methods. Unlike geometries, there are no builders for segments. Segments are fully defined when created, setting all properties at once.

The following principles apply when editing parts using segments:

- Use `Insert(int, Segment)` to insert a segment into a part at a specific index. This increases the number of segments in the part by one. Be careful: this will result in a gap at either end of the new segment if the start and end vertices do not share a location with the preceding and following adjacent segments, respectively.

```
// create a simple line segment
var segment = new LineSegment(-111.5, 37.5, -112.0, 38.0, SpatialReferences.Wgs84);
// insert it as the 2nd segment in the first part
builder.Parts[0].Insert(1, segment);
```

- Use `Remove(Segment)` or `RemoveAt(int)` to remove a segment from a part. If you remove anything other than the first or last segment in the part, this will result in a gap between the remaining segments. The number of segments decreases by one, and connections between segments are preserved.

```
// remove the third segment from the first part
builder.Parts[0].RemoveAt(2);
```

- Because segments are immutable, existing segments in a part cannot be changed. However, you can replace the existing segment at a specified index with a new segment by calling `RemoveAt` followed by `Insert`.

```
// remove the third segment from the first part
builder.Parts[0].RemoveAt(2);
// create a new line segment
var newSegment = new LineSegment(-111.5, 37.5, -112.0, 38.0, SpatialReferences.Wgs84);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// insert it as the new 3rd segment in the part
builder.Parts[0].Insert(2, segment);
```

- You do not need to close a part belonging to a polygon.

Caution:

When working with segment-based methods that take an `index` parameter, keep in mind that this is the index of the segment within the part and is a different index than that used when working with point-based helper methods. If a segment's ending vertex is at the same location as the starting vertex of the next segment, this location is represented by a single `MapPoint` in point-based methods. However, if there is a gap, separate `MapPoints` are returned for the two locations that define the gap.

Topological simplicity in features and graphics

To be saved as a feature in a geodatabase, a geometry must be simplified—that is, be topologically consistent. The rules vary for different types of geometry but include requirements such as polygons not having overlapping parts. Pass a geometry to the `GeometryEngine` object's `Simplify` method to return a simplified copy that can be saved in a geodatabase.

Note that in some cases, simplification can change the contents of the geometry, including resulting in an empty geometry. Topological simplicity is different than geometry generalization. For more information, see ITopologicalOperator.Simplify in the ArcObjects help.

The requirements for a geometry that will be used in a graphic are less rigorous than those used in a feature—there is no need to simplify geometries to display them in graphics.

Builders and IsValidSketch

In some workflows, both graphics and features are used. In interactive editing scenarios, for example, a user sketches a geometry on the map. A graphic is used initially to display the state of the sketch during sketching. When the user decides that the sketch is complete, the geometry can be saved to a geodatabase feature. You can use `IsValidSketch` on a geometry builder to determine when the builder contains enough state to be converted successfully to a geometry. This is a useful tool to enable or disable saving the sketch by the user. To be considered a valid sketch, a polygon builder must contain at least one part with three or more vertices in it, and a polyline builder must contain at least one part with at least two vertices. A multipoint builder must contain at least one point, and for a point or envelope builder, the x and y coordinates must not be NaN. If these conditions are satisfied, these cases should result in a geometry that could be visible if drawn.

Although `IsValidSketch` may be a good first check when building a geometry, it should not be relied upon to indicate if the eventual geometry will be valid or empty after simplification. Before allowing a user to save a geometry, consider taking a further verification step of calling `Simplify`. This approach can also be used for other situations that require geometries to be simple, for example, measurement or topological operations.

When importing or exporting geometries, for example, to JSON, if a multipart geometry has insufficient points, additional points can be added to ensure each segment has two endpoints to preserve the stability of other clients that may expect this.

Z-values and m-values

All types of geometry can have z-values. When building geometries, a simple rule determines whether or not the geometry stores z-values. If you call a method that has a z-value parameter, or add a point or segment that has z-values to a builder, the resulting geometry will store z-values—`HasZ` will return true.

Copyright © 1995-2020 Esri. All rights reserved.

Geometries can also have m-values, used in linear referencing workflows. Using m-values is generally less common than using z-values. Therefore, instead of using constructors, the `MapPoint` and `Envelope` classes present static factory methods that allow the creation of geometries with m-values.

Pay special attention to z- and m-values in editing workflows. When syncing edits, sync errors may occur if the z- or m-awareness of the geometry does not match that of the service. Find out more in the ArcGIS REST API help topic Error handling with sync m and z values.

Tip:

When saving geometries in geodatabase features, check that `HasZ` and `HasM` match the attribute awareness of the feature table being edited.

Related topics

Editing
Edit features
Sync offline edits
Geometries
Spatial references

Copyright © 1995-2020 Esri. All rights reserved.

Relate features

You may relate a feature in one table to a feature in another table for a variety of reasons, such as to allow your users to view information in one table for features in another. For example, you might want a user to be able to view fire hydrant inspection reports from a reports table, and view general fire hydrant information from a hydrants table - for the same set of fire hydrants.

When you relate features, you're editing them because the relate operation changes the attribute of the destination feature. The relate operation associates the two tables by setting up a keyField attribute in the destination feature based on the corresponding origin feature. The keyField value is the origin table feature's primary key or global ID. In this hydrant example, relating a hydrant and an inspection report adds the keyField value of the hydrant feature in the hydrants table (origin table) to the corresponding (hydrant) field in the report feature in the reports table (destination table).

The tables of the two features you want to relate must be in a table relationship with each other (must be related tables). Table relationships are created using ArcGIS Desktop.

The pattern for editing a feature to relate it to another is the same as for editing a "regular" feature, as described in Edit features, whether the feature is the origin feature in a table relationship or the destination feature. When tables participate in a table relationship like this, you can perform additional editing-oriented operations not described in Edit features but described on this page:

- Relate two features
- Unrelate features
- Validate relationship constraints

Relate two features

You use the relate method to relate a feature in one table to a feature in a different table as long as the two participating tables have a valid relationship between them. The relate method sets up the foreign key (`KeyField`) value on the destination feature and can be called on either the origin or destination feature.

Relating features is akin to changing the attributes of feature; it's synchronous and done in memory. A feature once related must be added or updated to the table using `AddFeatureAsync(feature)` or `UpdateFeatureAsync(feature)` for changes to be committed.

It's important to know the cardinality of the table relationship when you relate features.

The sample feature service at http://sampleserver6.arcgisonline.com/arcgis/rest/services/ServiceRequest/FeatureServer contains a layer of point features that represent customer requests for service. A related table contains zero or more comments for each feature in the service request layer. The following code uses this service to show how to add a new related comment to a service request feature clicked in the map.

```
// Identify a feature in the 'service request' layer (from a tap on the map, for example)
IdentifyLayerResult idResult = await MyMapView.IdentifyLayerAsync(serviceRequestLayer,
e.Position, 5, false);
ArcGISFeature serviceRequestFeature = idResult.GeoElements.FirstOrDefault() as ArcGISFeature;
if (serviceRequestFeature == null) { return; }

// Get the feature table from the feature
ArcGISFeatureTable serviceRequestTable = serviceRequestFeature.FeatureTable as
ArcGISFeatureTable;

// Get a list of tables related to the 'service requests' feature table
IReadOnlyList<ArcGISFeatureTable> relatedTables = serviceRequestTable.GetRelatedTables();
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Assume the comments table is the first (only?) related table
// (if there are many related tables, you can loop through the collection and check the table
name)
ArcGISFeatureTable commentsTable = relatedTables.FirstOrDefault();

// Create a new feature in the comments table
ArcGISFeature newComment = commentsTable.CreateFeature() as ArcGISFeature;

// Add comment text to the 'comments' attribute
newComment.Attributes["comments"] = "Please show up on time!";

// Relate the selected service request to the new comment
serviceRequestFeature.RelateFeature(newComment);
```

Unrelate features

You may want to unrelate two already related features. For example, when a feature has a constraint violation, such as an invalid cardinality, you may want to undo the last relate operation that caused the violation. Unrelating features resets the key field of the destination feature. Like the relate method, unrelate is also synchronous and is done in memory. Therefore `AddFeatureAsync(feature)` or `UpdateFeatureAsync(feature)` must be called for changes to be committed to the table.

The following code loops through related comments for a service request feature and removes those more than seven days old.

```
// Get the layer info from the service request table
ArcGISFeatureLayerInfo serviceRequestLayerInfo = serviceRequestTable.LayerInfo;

// Get all relationship infos
IReadOnlyList<RelationshipInfo> relationships = serviceRequestLayerInfo.RelationshipInfos;

// Find the info for the 'service request comments' relationship
var relatedCommentsInfo = (from r in relationships
                           where r.Name == "ss6_gdb_ServiceRequestComment"
                           select r).FirstOrDefault();

// Create a relationship query parameters using this relationship info
RelatedQueryParameters relatedCommentsQueryParams = new
RelatedQueryParameters(relatedCommentsInfo);

// Query the service requests table to get related comments
IReadOnlyList<RelatedFeatureQueryResult> relateResults = await
serviceRequestTable.QueryRelatedFeaturesAsync(serviceRequestFeature, relatedCommentsQueryParams);
if (relateResults.Count > 0)
{
    // Get the comments from the relationship results
    var relatedComments = relateResults.FirstOrDefault().ToArray();

    // Loop through all comments related to this service request
    foreach (ArcGISFeature relatedFeature in relatedComments)
    {
        // Load the related feature before accessing attribute values
        await relatedFeature.LoadAsync();

        // Get the date and time this comment was submitted
        DateTimeOffset submittedOn = (DateTimeOffset)relatedFeature.Attributes["submitdt"];

        // See if the comment is more than seven days old
        if (DateTime.Now.AddDays(-7) > submittedOn)
        {
            // Unrelate the comment from the service request (doesn't delete the related feature)
```

Copyright © 1995-2020 Esri. All rights reserved.

```
            serviceRequestFeature.UnrelateFeature(relatedFeature);
        }
    }

    // Update the feature to commit changes to the table
    await serviceRequestTable.UpdateFeatureAsync(serviceRequestFeature);
}
```

Validate the relationship constraints of a feature

You can check if a given feature violates any relationship constraints (for example, when a hydrant inspection report is being added by a user), such as cardinality violations. In 1:1 relationships, if an origin feature is already related to a destination feature, no other destination feature can be related to it. In 1:n relationships, a destination feature can be related to only one origin feature while an origin feature can be related to one or more destination features. For more info on cardinality and types of relationships, see Relationship class properties in the ArcGIS Desktop help.

The following branching statement checks for a relationship constraint violation for the service request feature and its related comment records.

```
// See if this service request feature violates relationship constraints with comments
RelationshipConstraintViolationType constraintViolation = await
commentsTable.ValidateRelationshipConstraintsAsync(serviceRequestFeature);

// Check for a constraint violations
if (constraintViolation == RelationshipConstraintViolationType.Cardinality)
{
    // Cardinality means the wrong number of related features (more than 1 related record in a
1:1 relationship, for example)
}
else if (constraintViolation == RelationshipConstraintViolationType.Orphaned)
{
    // Orphaned means a record with no related feature when one is required
}
else
{
    // No violations
}
```

Note that an update operation on a feature succeeds even if a relationship constraint has been violated by the edit. This is consistent handling with ArcGIS Server and ArcGIS Desktop, which do not enforce cardinality violations. ArcGIS Desktop provides a **Validate Feature** tool, which allows you to recover from violations in a back-office operation after applying edits or syncing, if you choose to do so. See Validating features and relationships in ArcMap for more information. However, if you wish to check for violations in your app and recover from them, you can use the validate method.

Another type of constraint violation captured by the validate method occurs in a composite relationship when an orphan feature is added to the destination table without relating it to an origin feature. You can recover from this violation by relating the orphaned destination feature to a valid origin feature.

Tip:

Performance tip: If the related features are not local to the device, the validate method makes network calls to query for them. As a result, it can negatively impact performance when used on a number of features. This hit to performance is why edit operations such as add, update, and delete do not check proactively for relationship constraint violations.

Copyright © 1995-2020 Esri. All rights reserved.

Additional information

Getting related tables for a layer or table on the map returns only those tables on the map. Similarly, related queries require both origin and destination table/layer to be present on the map. For tables not on the map, you can query them using regular query operations but cannot use related queries. You can also view what relationships the tables participate in.

All the tables participating in a relationship must be present in the data source. ArcGIS Runtime supports related tables in the following data sources:

- ArcGIS feature service
- ArcGIS map service
- Geodatabase downloaded from a feature service
- Geodatabase in a mobile map package

Two participating tables can have one of the following cardinalities: one-to-one, one-to-many, or many-to-many.

When defining relationships, one table must be origin and the other destination. A table can participate in more than one relationship. A table can play a destination role in one relationship and origin in another, resulting in nested relationships.

Simple and composite workflow-based relationships are supported:

- In a simple relationship, features in the destination table are independent to features in the origin table. For example, a transformer and an electric pole may be related but they can also exist their own. Deleting an origin feature resets the `keyField` of the relationship to NULL in the destination feature.
- In a composite relationship, each feature in the destination table is expected to be related to an origin feature. In other words, any orphan destination features are considered a violation of the relationship. For example, a building and its mailbox must be related. While the building (origin) can exist on its own, a mailbox (destination) must be related to a building. When an origin feature is deleted, the destination feature should also be deleted. This is known as a cascade delete.

Copyright © 1995-2020 Esri. All rights reserved.

Use geodatabase transactions

A **transaction** manages a series of geodatabase edits as a single unit of work. If all edits in a transaction are successful, it can be **committed** and saved to the geodatabase. If a transaction encounters errors and must be canceled, it is **rolled back** to discard all edits. ArcGIS Runtime provides an API to manage your geodatabase edits using transactions and to commit or roll them back as appropriate.

Note:
While the use of transactions is available for editing local geodatabases, it is not required. If it works better for your workflow, you can continue to make edits directly to your data without the use of transactions.

The following steps describe a general workflow for using transactions to manage local geodatabase edits.

1. **Start a transaction:** You can start a transaction by calling `BeginTransaction` on the geodatabase that contains the data you want to edit. The transaction stays active until you end it either by committing or rolling back. Use `IsInTransaction` to check whether or not the geodatabase has a current transaction.

```
// see if there is a transaction active for the geodatabase
if (!_localGeodatabase.IsInTransaction)
{
    // if not, begin a transaction
    _localGeodatabase.BeginTransaction();
}
```

   When a transaction is active, the geodatabase is locked for editing. ArcGIS Runtime uses the **immediate** transaction mode supported by SQLite. For details, see Database locking below.

2. **Make edits:** Add or delete features, update feature geometry and attributes, edit feature attachments and related features, and so on. All edits you make will be included in the current transaction. You can include individual edits as well as bulk edits (inserting a set of features, for example). You cannot synchronize the local geodatabase with a service (geodatabase sync uses its own transaction). You cannot begin another transaction (nested transactions are not supported).

3. **End the transaction**: End a transaction by committing or rolling back edits. If you want to make additional edits and manage them in a transaction, you need to call `BeginTransaction` to start a new one.

   a. **Commit edits:** To store all edits made in the transaction, call `CommitTransaction`.

```
// see if there is a transaction active for the geodatabase
if (_localGeodatabase.IsInTransaction)
{
    // if there is, commit the transaction (this will also end it)
    _localGeodatabase.CommitTransaction();
}
```

   b. **Rollback edits:** To discard all edits and end the transaction, call `RollbackTransaction`.

```
// see if there is a transaction active for the geodatabase
if (_localGeodatabase.IsInTransaction)
{
    // if there is, rollback the transaction to discard the edits (this will also end the
transaction)
    _localGeodatabase.RollbackTransaction();
}
```

Copyright © 1995-2020 Esri. All rights reserved.

4. **Handle changes in transaction state (optional):** You can handle the `TransactionStatusChanged` event on the geodatabase to listen for changes in the transaction state. This event will notify when geodatabase transactions begin or end. You might use this information to enable and disable editing controls as appropriate.

```
// handle transaction status change
_localGeodatabase.TransactionStatusChanged += (s, e) =>
{
    // check the new transaction status
    if (e.IsInTransaction)
    {
        // handle transaction starting ...
    }
    else
    {
        // handle transaction completing ...
    }
    };
```

Database locking

ArcGIS Runtime uses the **immediate** transaction mode supported by SQLite. When you begin a transaction, the geodatabase will be locked for writing. Others can read the geodatabase, but they will not be able to write or to see your uncommitted edits. The immediate transaction mode allows only one transaction at a time, so attempting to begin a transaction when one is active will result in an error. Use the `IsInTransaction` property on the geodatabase to see if there is a current transaction.

ArcGIS Runtime starts a transaction internally when synchronizing geodatabase edits to a service. Therefore, if you attempt to begin a transaction during synchronization, you will receive an error that states a transaction is already in progress. Likewise, if you attempt to synchronize the database inside your own transaction, you will receive the same error.

Commit or rollback to end a transaction

A geodatabase transaction stays active until you end it by calling `CommitTransaction` or `RollbackTransaction`. All edits made between the `BeginTransaction` call that started the transaction and this call are part of the transaction. `CommitTransaction` saves all edits in the transaction to the geodatabase and ends the transaction. `RollbackTransaction` discards all edits in the transaction and ends the transaction.

⚠️ **Caution:**

Calling commit or rollback without a current transaction will result in an exception.

Transactions allow you to manage a set of edits as a single unit. This may be required when working with edits that must either succeed or fail as a group. This is similar to a financial transaction, where you might want to deposit a check and withdraw part of it. If the check cannot be deposited, you should not attempt to make the withdrawal. Similarly, if any edit that's part of a geodatabase transaction fails, you can rollback all edits.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
ArcGIS Runtime internally manages some operations within a transaction. Synchronizing a geodatabase with a service or making bulk edits (inserting a set of features, for example) are both handled in a transaction that will be rolled back if any of the edits fail.

If you're unexpectedly disconnected from the database, any active transaction will be rolled back. If the app crashes, or the user closes the app without saving edits, for example, the transaction will be rolled back and the edits discarded.

Copyright © 1995-2020 Esri. All rights reserved.

Sync offline edits

Your users can edit offline and sync their edits to a feature service later when connectivity is restored. Syncing offline edits requires that a local geodatabase is created from an existing sync-enabled feature service either from ArcGIS Online or ArcGIS Enterprise. A user can take a single sync-enabled feature service offline or a map containing one or more sync-enabled feature services offline.

If you take a single feature service offline and edit the data in the geodatabase, you can sync the single geodatabase with the feature service using the `GeodatabaseSyncTask` (see Sync offline edits.) Alternatively, if you take a whole map offline and create a geodatabase for each of the sync-enabled feature services in the map, you can sync all the geodatabases using the `OfflineMapSyncTask` (see Synchronize a map.)

You should note that syncing can be performed, even if no edits have been made locally, to pull changes from the feature service into the local copy of the data.

Note:
A Basic license is required for editing.

Synchronize a single geodatabase

To synchronize edits made to a single geodatabase, do the following:

- Define synchronization parameters, such as the direction in which edits should be synchronized.
- Create a new `GeodatabaseSyncTask` that points to the URL of the feature service.
- Call `SyncGeodatabase` on the task to create a new `SyncGeodatabaseJob`. Pass in the synchronization parameters and the local geodatabase.
- Handle the `JobChanged` event to report status and messages as the job runs.
- Await the asynchronous completion of the job.

```
// function to submit a geodatabase synchronization job
// the URL for the feature service and the path to the local geodatabase are passed in
public async Task SyncronizeEditsAsync(string serviceUrl, string geodatabasePath)
{
    // create sync parameters
    var taskParameters = new SyncGeodatabaseParameters()
    {
        RollbackOnFailure = true,
        GeodatabaseSyncDirection = SyncDirection.Bidirectional
    };

    // create a sync task with the URL of the feature service to sync
    var syncTask = await GeodatabaseSyncTask.CreateAsync(new Uri(serviceUrl));

    // open the local geodatabase
    var gdb = await Esri.ArcGISRuntime.Data.Geodatabase.OpenAsync(geodatabasePath);

    // create a synchronize geodatabase job, pass in the parameters and the geodatabase
    SyncGeodatabaseJob job = syncTask.SyncGeodatabase(taskParameters, gdb);

    // handle the JobChanged event for the job
    job.JobChanged += (s, e) =>
    {
```

Copyright © 1995-2020 Esri. All rights reserved.

```
        // report changes in the job status
        if (job.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Succeeded)
        {
            // report success ...
            statusMessage = "Synchronization is complete!";
        }
        else if (job.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Failed)
        {
            // report failure ...
            statusMessage = job.Error.Message;
        }
        else
        {
            statusMessage = "Sync in progress ...";
        }
    };

    // await the completion of the job
    var result = await job.GetResultAsync();
}
```

Note:
The sync operation overwrites previously synced edits to the same features on the service.

Internally, ArcGIS Runtime uses a geodatabase transaction to wrap the synchronization process. Since only one current transaction is permitted on a geodatabase, you will receive an error if you attempt to synchronize while another transaction is active (in an editing workflow, for example). Similarly, you may get an error if you try to start a transaction while synchronization is in progress. Errors that arise during a sync operation are returned in the callback when the job is done. For descriptions of errors that can arise when syncing offline edits, see Error handling with sync.

For services backed by non-versioned data, sync operations are performed per-layer, and are always bi-directional—that is, local edits are uploaded to the service, and then new edits are downloaded from the service. For services backed by versioned data, sync operations are per-geodatabase, and you can change the synchronization parameters to determine in which direction edits are synchronized—download only, upload only, or bi-directional. Use `Geodatabase.SyncModel` to find out if a geodatabase can be synchronized per-layer or per-geodatabase. Use `SyncGeodatabaseParameters.GeodatabaseSyncDirection` to set the synchronization direction for a sync operation. When using bi-directional sync, note that the 'last in wins'—that is, uploaded edits will overwrite changes present on the server.

Synchronize a map

You can synchronize all the edits made to all the sync-enabled geodatabases in your map using the offline map sync task.

- Create an `OfflineMapSyncTask`.
- Create an `OfflineMapSyncParameters` object with appropriate values.
- Create an `OfflineMapSyncJob` by calling `SyncOfflineMap` on the task, passing in the parameters object you defined.
- Handle the `OfflineMapSyncJob.JobChanged` event to track status changes and messages for the job. This is where job completion (success or failure) will be reported.
- Optionally handle the `OfflineMapSyncJob.ProgressChanged` event to monitor and report the job's progress.
- Call `OfflineMapSyncJob.Start` to start the sync job.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create a new offline map sync task
OfflineMapSyncTask offlineMapSyncTask = await OfflineMapSyncTask.CreateAsync(map);

// Create the offline map sync parameters
OfflineMapSyncParameters parameters = new OfflineMapSyncParameters
{
    RollbackOnFailure = true,
    SyncDirection = SyncDirection.Bidirectional
};

// Instantiate the sync job using the synchronization parameters
OfflineMapSyncJob offlineMapSyncJob = offlineMapSyncTask.SyncOfflineMap(parameters);

// Add listener to deal with the completed job
offlineMapSyncJob.JobChanged += (s, e) =>
{
    // report changes in the job status
    if (offlineMapSyncJob.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Succeeded)
    {
        // report success ...
        statusMessage = "Synchronization is complete!";
    }
    else if (offlineMapSyncJob.Status == Esri.ArcGISRuntime.Tasks.JobStatus.Failed)
    {
        // report failure ...
        statusMessage = offlineMapSyncJob.Error.Message;
    }
    else
    {
        statusMessage = "Sync in progress ...";
    }
};

// Start the job
offlineMapSyncJob.Start();
```

Register a geodatabase workflow

To work offline with a sync-enabled feature service, you need to download a geodatabase containing the features to your device. As mentioned in the Take a layer offline, Take a map offline - on-demand, and Take a map offline - preplanned topics, you can download a geodatabase to your device using any of these methods:

- `GenerateGeodatabase` on `GeodatabaseSyncTask`
- `GenerateOfflineMap` on `OfflineMapTask`
- `DownloadPreplannedOfflineMap` on `OfflineMapTask`

Each of these methods will automatically register the geodatabase with the feature service so that you can sync any changes with the feature service as described above.

If, however, you copy that geodatabase file to another device, the feature service will not be able to perform a sync with that geodatabase. If you wish to enable sync you must register the geodatabase with the feature service you used to generate the original geodatabase. To do this you must use the use `RegisterGeodatabaseAsync` method on `GeodatabaseSyncTask` to register each geodatabase copy (on each device). Registering in this way ensures each device receives the correct updates during sync operations.

Copyright © 1995-2020 Esri. All rights reserved.

Caution:

- Registering a geodatabase with a feature service is not supported with a versioned feature service.
- Once you call unregister on a geodatabase, you cannot re-register the same geodatabase.
- If the original geodatabase is ever unregistered, no additional clients can use that copy to register.

Related topics

Editing
Edit features
Edit geometries

Copyright © 1995-2020 Esri. All rights reserved.

Edit KML content

KML, short for Keyhole Markup Language, is a standard that defines an XML-based format for representing geographic features and map visualizations. You can create and edit KML content in your ArcGIS Runtime app. ArcGIS Runtime saves KML version 2.2 to a compressed KML format known as KMZ.

A KML dataset is a container for a hierarchical collection of KML nodes. Some common types of KML nodes include the following:

- Document—A container for KML features (placemarks, overlays, folders, and so on).
- Folder—A container that defines a hierarchy for nodes in the tree.
- Ground overlay—Specifies a local or hosted image file to draw on the surface.
- Model—Represents an object in 3D coordinate space using a COLLADA file.
- Network link— Specifies the location of a KML or KMZ resource on a local or remote network.
- Placemark—Similar to an ArcGIS feature, it represents a geographic location with point, line, or polygon geometry.
- Screen overlay—Specifies a local or hosted image file to draw on the screen (such as a company logo).
- Tour—Defines a fly-through experience composed of an ordered list of elements to visit.
- Track/Multitrack—Describes the geographic position of an object over time.

ArcGIS Runtime SDK supports the creation and editing of documents, folders, placemarks, ground overlays, screen overlays, and network links. You can also edit the properties of KML nodes, including the name, description, snippet, geometries, and style. KML content you create or edit can be saved to a `.kmz` file.

Some advanced content types or behaviors are currently not supported for editing via ArcGIS Runtime, such as the following:

- Shared style elements
- Highlight style

Note:
KML geometry always uses the WGS84 coordinate system.

Create KML content and save it to a KMZ file

You can save any KML node to an output file by calling `SaveAsAsync`. Doing so writes the KML representation of the node and all of its child nodes to a compressed KML format (`.kmz`). Only files referenced by absolute URLs are supported for use as icons and links. It's common to save a KML document node since it serves as a top-level container for several related child nodes. The output KMZ archive is shared easily as a single file and can be read into any app that supports KML, such as ArcGIS Earth or an ArcGIS Runtime SDK app. See Display KML content for details.

The following steps describe how to create a KML dataset, add nodes, and save it locally to a new KMZ file.

1. Create a KML document and use it to create a KML dataset. The document is a top-level container for additional KML nodes.

```
// Create a new KML document and set its name property.
KmlDocument kmlDocument = new KmlDocument
{
    Name = "UFO Sightings"
};
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create a new KML dataset using the KML document.
KmlDataset kmlDataset = new KmlDataset(kmlDocument);
```

2. Create a style to represent a placemark. The style defines an icon style that is created with an icon (built with a URI to a local or online image).

```
// Create a new icon using a URL to an online image.
KmlIcon placemarkIcon = new KmlIcon(new System.Uri("http://static.arcgis.com/images/Symbols/
SafetyHealth/esriCrimeMarker_56_Gradient.png"));

// Create an icon style using the icon, adjust the size using the scale argument.
KmlIconStyle iconStyle = new KmlIconStyle(placemarkIcon, 3);

// Create a KML style that uses the icon style.
KmlStyle placemarkStyle = new KmlStyle
{
    IconStyle = iconStyle
};
```

3. Create a placemark node with point geometry and set its style to use the icon. Provide a value for the ID, Name, and Description to give more information about the placemark.

```
// Create a KML placemark, give it a name and description.
MapPoint mapPoint = new MapPoint(-105.243, 33.968, 0.0, SpatialReferences.Wgs84);
KmlGeometry kmlGeometry = new KmlGeometry(mapPoint, KmlAltitudeMode.ClampToGround);
KmlPlacemark kmlPlacemark = new KmlPlacemark(kmlGeometry)
{
    Id = "43a",
    Name = "Roswell crash site",
    Description = "Debris discovered at a ranch in Roswell New Mexico in 1947 is widely " +
                    " believed to be from an alien spacecraft."
};

// Apply the icon style to the placemark.
kmlPlacemark.Style = placemarkStyle;

// Add the placemark to the KML node collection.
kmlDocument.ChildNodes.Add(kmlPlacemark);
```

4. Save the KML document as a `.kmz` file. You can save any KML node to write its contents (its child nodes, in other words) to a file on disk.

```
// Save the KML document as a local .kmz file.
await kmlDocument.SaveAsAsync(outFolder + "TruthIsOutThere.kmz");
```

5. Optionally, display the new KML file as a KML layer in your map or scene.

```
// Create a new KML layer from the output KMZ file.
KmlLayer kmlLayer = new KmlLayer(new System.Uri(outFolder + "TruthIsOutThere.kmz"));

// Add the KML layer to the scene (or map).
MySceneView.Scene.OperationalLayers.Add(kmlLayer);

// Zoom to the extent of the new layer.
```

Copyright © 1995-2020 Esri. All rights reserved.

```
Envelope layerExtent = kmlLayer.FullExtent;
await MySceneView.SetViewpointAsync(new Viewpoint(layerExtent));
```

Create a KML ground overlay

A ground overlay is a KML element that displays an area on the surface. Like a placemark node, a ground overlay is created with geometry and a symbol. It has name, description, and snippet properties that can be used to display additional information. Geometry for a ground overlay must be of type polygon or envelope. The data the overlay represents comes from an image reference in a KML icon.

The following steps illustrate creating a simple ground overlay and adding it to a KML document's node collection:

1. Create an icon to display with the ground overlay. The icon is defined using a URI to an online or local image.

```
// Create an icon from a image URL to display the ground overlay.
KmlIcon icon = new KmlIcon(new System.Uri("http://static.arcgis.com/images/Symbols/Basic/
CrossHair.png"));
```

2. Create geometry to use for the ground overlay. The ground overlay geometry must be a polygon or an envelope.

```
// Use a polygon builder to create a rectangle polygon (in WGS 84 coordinates).
PolygonBuilder overlayPolyBldr = new PolygonBuilder(SpatialReferences.Wgs84);
overlayPolyBldr.AddPoint(-115.858, 37.204);
overlayPolyBldr.AddPoint(-115.858, 37.326);
overlayPolyBldr.AddPoint(-115.757, 37.326);
overlayPolyBldr.AddPoint(-115.757, 37.204);
Polygon overlayPoly = overlayPolyBldr.ToGeometry();
```

3. Create a KML ground overlay with the geometry and icon to use. Add the new overlay to the KML document's child node collection.

```
// Create a new KmlGroundOverlay with the polygon and icon.
KmlGroundOverlay groundOverlay = new KmlGroundOverlay(overlayPoly, icon);

// Provide a name and description.
groundOverlay.Name = "Area 51";
groundOverlay.Description = "A highly classified US Air Force facility in the Nevada Test and
```

Copyright © 1995-2020 Esri. All rights reserved.

```
Training Range.";

// Add the ground overlay to the document's child nodes.
kmlDocument.ChildNodes.Add(groundOverlay);
```

Create a KML screen overlay

A screen overlay is an image shown on top of the map, with a fixed position relative to the screen. Screen overlays are commonly used for compasses, logos, legends, and heads-up displays. Screen overlays are defined in screen coordinates and appear in the same location on the display when the geoview is panned or zoomed.

The following steps illustrate creating a simple screen overlay and adding it to a KML document's node collection:

A screen overlay is a KML element that displays an image on the screen. Like a ground overlay node, a screen overlay is created with geometry and a symbol. Unlike nodes that display on the map, the geometry for a screen overlay is defined relative to the screen, in display units such as pixels, or as a fraction of the width and height of the display. The overlay image is defined using a KML icon.

The following steps illustrate creating a north arrow screen overlay and adding it to a KML document's node collection:

1. Create an icon to display with the screen overlay. The icon is defined using a URI to an online or local image.

```
// Create a north arrow icon from a image URL to display in the screen overlay.
KmlIcon northIcon = new KmlIcon(new System.Uri("http://static.arcgis.com/images/Symbols/
Cartographic/esriCartographyMarker_47.png"));
```

2. Create a KML image coordinate to define the location for the screen overlay. The screen overlay location is defined relative to the screen display. This can be defined in coordinates, such as pixels, or as a fraction of the width and height of the display.

```
// Define the screen coordinate (in pixels) where the overlay will display.
KmlImageCoordinate northArrowLocation = new KmlImageCoordinate(100, 100, KmlUnitsType.Pixels,
KmlUnitsType.Pixels);
```

Note:
The overlay location (screen coordinate) is mapped to a point on the icon. By default, this is the center of the image. This point can be explicitly defined by setting the `OverlayCoordinate`.

3. Create a KML screen overlay and define the location and icon. Add the new overlay to the KML document's child node collection.

```
// Create a new KmlScreenOverlay with the north arrow icon.
KmlScreenOverlay screenOverlay = new KmlScreenOverlay(northIcon);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Set the screen overlay display coordinate.
screenOverlay.ScreenCoordinate = northArrowLocation;
  // Add the screen overlay to the document's child nodes.
kmlDocument.ChildNodes.Add(screenOverlay);
```

Copyright © 1995-2020 Esri. All rights reserved.

Route and get directions

Copyright © 1995-2020 Esri. All rights reserved.

Find a route

With ArcGIS Runtime SDK, you can:

- Calculate point-to-point and multipoint routes
- Optimize the results to find the shortest or the fastest route
- Reorder stops to find the best sequence
- Avoid restricted areas and maneuvers
- Specify time windows of arrival and departure for each stop

A route task is a network analysis task that is executed asynchronously. It returns results with details about a route that visits two or more stops (locations) within a transportation network. This operation is sometimes referred to as solving a route.

Transportation networks are created from features (lines and points) that represent roads, bridges, tunnels, bike paths, train tracks and other elements in the network. The geometric intersections of features help to define connectivity between the network elements they represent. The connectivity of the transportation network is analyzed to find a route.

Note:
To learn more, see the topic What is a network dataset.

This topic's first sections lead through the process to find a route using an online service. See the section Routing with local data to see how using local data is different from using a service. Whether you use a route service or local data, the overall process is the same.

Choosing a routing data source

You can use a transportation network exposed as an online ArcGIS route service (an ArcGIS Network Analysis service). To use an online service, you must have a network connection to the service from your device.

This topic's first sections lead through the process to find a route using an online service. See the section Routing with local data to see how using local data is different from using a service. Whether you use a route service or local data, the overall process is the same.

Network Analyst services are hosted in Esri's cloud platform, ArcGIS Online, or can be published on your own ArcGIS servers. These services provide a REST API for clients such as mobile and Web applications. For more information about publishing a service, see Tutorial: Publishing a network analysis service.

How do route services differ? The most important criteria for choosing a service (or a transportation network for that matter) is the area that it covers. You cannot find a route using a service that does not encompass all the stops and barriers that your user will specify. In addition, route services offer differing levels of capability and specialization.

You can build a transportation network from your own data, or use a dataset provided by someone else, such as Esri's World Street Map data. Then, you can publish a route service on your own portal or on ArcGIS Online. Or, you could use a route service created by someone else that suits your needs. For example, the Esri World Route Service is capable of finding routes almost anywhere because it references road data that covers much of the world. Since the ArcGIS Online road data also includes traffic data for many areas, you can find the best routes and account for dynamic traffic conditions. For instance, you can find the best route from home to work with a departure time of 7:00 a.m. and compare that route with one you generate for 8:00 a.m.

Copyright © 1995-2020 Esri. All rights reserved.

Like other services, some route services require authentication to access. You provide this authentication when you create the route task object.

Creating and loading the route task

The route task is an object that refers to the transportation network, accepts route task parameters, is executed to find a route, and reports results.

```
var routeSourceUri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/
NetworkAnalysis/SanDiego/NAServer/Route");
var routeTask = await RouteTask.CreateAsync(routeSourceUri);
```

Specifying route task parameters

Route task parameters specify how a route should be found. There are many parameters that let you customize how the route is determined. Parameters include stops and barriers. (More about those later.) Other parameters include travel mode, whether stops map be reordered to find an optimized route, the length units to be used in the turn-by-turn directions, and so forth.

The best way to specify parameters is to call the method on the route task that retrieves the default routing parameters defined for the service, and then change the defaults as needed before executing the route task. The service's default parameters support the most common use case for that service. Different services can have different defaults. While it is possible to create the parameters from scratch, this approach is prone to errors and usually not done.

```
// get the default route parameters
var routeParams = await routeTask.CreateDefaultParametersAsync();
// explicitly set values for some params
routeParams.ReturnDirections = true;
routeParams.ReturnRoutes = true;
routeParams.OutputSpatialReference = MyMapView.SpatialReference;
```

Your app might allow users to choose some parameters directly, such as whether the stops should be followed in the order specified or in optimum order. Your app would get these choices from the user and set the corresponding route task parameters.

Specifying stops and barriers

An optimal route could be from a start point to an end point. In the parlance of routing, both of these points are referred to as **stops**. A route must have at least two stops, but may have more. Although there are a variety of stop types, they are always point locations along a transportation network.

The role a stop serves in the route is defined with the `StopType`. A stop can be a `Stop` (the default), `Waypoint` or `RestBreak`. A `Stop` is a location that the route must visit. A `Waypoint` is location that the route must travel through without making a stop (for this reason, waypoint locations are not described in the driving directions). At a `RestBreak`, the route stops for the driver to take a break.

If you use a table to define stops, an integer field called `LocationType` can be used to define the stop type, where a **0** defines a `Stop`, **1** defines a `Waypoint`, and **2** defines a `RestBreak`.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
There are some restrictions for how waypoints and rest breaks can be used. Waypoints cannot be the first or last stop, have time windows defined, or have added costs defined. Waypoints and rest breaks cannot be used if `FindBestSequence` for the route is true.

How you get stops for the route is a design decision. Here are some options, and you'll probably come up with other ideas, too.

- Get the current location of your device and use it as a starting point.If your app shows the device's location, you may wish to use the geometry of this location as the starting point.
- Let users choose stops by clicking near roads on a map.
- Access an address from an address book, geocode the address, and use the result as a stop.

Barriers are places that your route must avoid because they are impassable, or places that add cost to the route if traversed. The best route usually has the least cost.

Unlike stops which are always points, barriers can be points, polylines, or polygons. Like stops, you can define barriers in various ways. Your app could allow users to draw barrier graphics. Another option is to use graphics from a feature table. For example, you may have a feature table with areas affected by a forest fire, and you want your route to avoid those areas.

You do not have to specify any barriers to find a route.

Regardless of how you define your route's stops and barriers, all must share a spatial reference. That spatial reference doesn't have to match the spatial reference of the route service or transportation network.

```
// create a Stop for my location
var myLocation = new MapPoint(-13034379.7137452, 3858390.14190331,
SpatialReferences.WebMercator);
var stop1 = new Stop(myLocation);

// create a Stop for your location
var yourLocation = new MapPoint(-13032763.2362459, 3860880.60006008,
SpatialReferences.WebMercator);
var stop2 = new Stop(yourLocation);

// assign the stops to the route parameters
var stopPoints = new List<Stop> { stop1, stop2 };
routeParams.SetStops(stopPoints);

// create a PointBarrier to avoid
var accidentLocation = new MapPoint(-13033331.7103904, 3859368.53617918,
SpatialReferences.WebMercator);
var barrier = new PointBarrier(accidentLocation);

// add the point barriers to the route parameters
List<PointBarrier> pointBarriers = new List<PointBarrier> { barrier };
routeParams.SetPointBarriers(pointBarriers);
```

Executing the route task

Execute the route task while providing the route task parameters.

```
var routeResult = await routeTask.SolveRouteAsync(routeParams);
// ... process route results ...
```

Copyright © 1995-2020 Esri. All rights reserved.

Processing results

The route task signals when the task is complete and results are available. The most familiar part of the results may be the linear graphic that traces the path of the route. Technically, it is a polyline that follows connected edges in the transportation network from stop to stop in a specific order. The route task can return the stops and barriers with the results. This makes it easier for your app to access all the associated graphics when displaying results. The route task also returns turn-by-turn directions that describe the route. If your app does not need one or more of these result elements, each can be turned off by setting route task parameters.

It is up to your app to present results in a useful way. Most commonly, the route, the stops, and sometimes the barriers are displayed on a graphics overlay on the map. The directions may be displayed in a list form or through a directions browser implemented by the app.

```
// get the route from the results
var route = routeResult.Routes[0];

// create a graphic (with a dashed line symbol) to represent the route
var routeSymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Dash, Color.Yellow, 5);
var routeGraphic = new Graphic(route.RouteGeometry, routeSymbol);

// add the route graphic to the map view and zoom to its extent
MyMapView.GraphicsOverlays[0].Graphics.Add(routeGraphic);
await MyMapView.SetViewpointGeometryAsync(route.RouteGeometry, 30);
```

Directions

Directions are returned for each route in the results if requested in the task parameters. Depending on the service configuration, directions can be returned in more than one language. Supported languages for a service are available in the closest facility task info.

Copyright © 1995-2020 Esri. All rights reserved.

Directions are composed of direction maneuvers. Each maneuver contains properties such as the direction text, which contains instructions that describe travel through a section of a route. A direction maneuver is further broken down into maneuver messages that give details about a specific maneuver.

Driving Directions

1: Start at Incident A
2: Go south on 6TH AVE toward BEECH ST
3: Turn right on BEECH ST
4: Turn right on 5TH AVE
5: Turn right onto ramp and go on I-5 S
6: Bear right onto ramp to PERSHING DR toward B STREET
7: At fork keep left on PERSHING DR
8: Continue on PERSHING DR
9: Turn left on FLORIDA DR
10: Turn right on MORLEY FIELD DR
11: Continue on ALABAMA ST
12: Finish at Facility X

If defined in the transportation network, directions may also include direction events, which are notifications that do not require any action when traversing the route, for example, crossing over a time zone boundary.

Direction maneuver's can include **from level** and **to level** values to define things like floor levels inside a building. For example, a user may want to see just one level of a walking route that spans multiple floors.

In order to return from level and to level values for a route:

- The route's directions style must be `Campus`
- The network dataset must be configured to support levels as described in the Setting directions topic in the desktop help. This applies to both local data and the underlying network used by a routing service
- The ArcGIS Server version hosting the service must be 10.6 or higher (when using online routing services)

Routing with local transportation network data

You can find a route using local transportation networks stored directly on your device. You can create these using ArcGIS Pro and deploy them to your device using either a mobile map package file (`.mmpk`), a mobile scene package file (`.mspk`), or a mobile geodatabase file.

If you'd like a ready-to-use and regularly updated network dataset (and locator) for your area of interest, you can license StreetMap Premium data (in mobile map package format). For details, see Add StreetMap Premium data.

Mobile map and scene packages

You can create a mobile map package (.mmpk) or create a mobile scene package (.mspk) using ArcGIS Pro. Each map or scene in the package can support a collection of transportation networks. Any of these can be used to construct the route task. Here is the general workflow to access one of these transportation networks:

1. Load the mobile package (.mmpk or .mspk) from local storage.
2. Get the desired map or scene from the package.
3. Retrieve the collection of transportation networks associated with that map or scene.

Copyright © 1995-2020 Esri. All rights reserved.

4. Construct and load a route task from one of these transportation networks.
5. Specify the route parameters and find a route.

```
// locate the mobile map package
MobileMapPackage mobileMapPackage = await MobileMapPackage.OpenAsync(mmpkPath);

// get the first map from the map package
Map streetMap = mobileMapPackage.Maps.FirstOrDefault();
if (streetMap != null)
{
    // get the first transportation network in the map
    TransportationNetworkDataset transportationNetwork =
streetMap.TransportationNetworks.FirstOrDefault();
          // instantiate a RouteTask with the network dataset
    RouteTask localRouteTask = await RouteTask.CreateAsync(transportationNetwork);
          // .. solve a route ...
}
```

Mobile geodatabase

When a transportation network is in a mobile geodatabase, you can access that network by specifying the location of the geodatabase and the name of the network.

Note:
To learn more, see Creating ArcGIS Runtime content.

```
var routeTaskLocal = await RouteTask.CreateAsync(geodatabasePath, networkDatasetName);
```

Related topics

Search for places (geocoding)

Copyright © 1995-2020 Esri. All rights reserved.

Display driving directions

Whether your ArcGIS Runtime app is connected or disconnected, your users can get driving directions in the form of a route on a map, turn-by-turn instructions, or both. Driving directions can be for a route between two locations (known as stops) or they can be for a number of other route types, such as:

- A multiple stop route—A route from one location to multiple stops. For credit usage calculation (ArcGIS Online), this route is known as **Simple Directions** unless you want the algorithm to re-sequence the stops in the optimal order (shortest distance or shortest time), in which case the route is known as **Optimized Directions**. You may know these optimized directions as the traveling salesman problem.
- A route with time windows—A route to some stops that have specific times of the day they need to be reached by the driver.
- A route with barriers—A route that prevents traversal of some streets due to barriers (for example, street closures) or that reduces the traversal speed of some streets.
- Drive time—Drive time refers to calculating the area that can be reached within a specified travel time or travel distance along a street network based on travel mode.

Tip:

If you'd like a ready-to-use and regularly updated network dataset (and locator) for performing offline geocoding and routing, you can license StreetMap Premium data (in mobile map package format). For details, see Add StreetMap Premium data.

Displaying driving directions is just one of many capabilities known as network analysis capabilities. Other network analysis capabilities include origin-destination (OD) cost matrix and location-allocation, which your app can consume through a geoprocessing service via a `GeoprocessingTask`.

To display driving directions, you typically follow these steps:

1. Create a `RouteTask` and `RouteParameters` that you'll use to reference the routing service.

   The code below sets up the route task and starts to define the parameters to be used to solve the route.

```
// Create a new RouteTask that uses the San Diego route service
var routeSourceUri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/
NetworkAnalysis/SanDiego/NAServer/Route");
var routeTask = await RouteTask.CreateAsync(routeSourceUri);

// Get the default route parameters
var routeParams = await routeTask.CreateDefaultParametersAsync();

// Set the output spatial reference for task results to match the map view's
routeParams.OutputSpatialReference = MyMapView.SpatialReference;

// Modify the default parameters to return driving directions and stops with the results
routeParams.ReturnDirections = true;
routeParams.ReturnRoutes = true;
```

2. Create the stops. If you want users to be able to type in an address or place name, then either:
   - Use a service that includes a locator, such as `http://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer`.

Copyright © 1995-2020 Esri. All rights reserved.

- Set up your own address locator using ArcGIS Desktop. If you set up your own address locator, you can serve it via ArcGIS Online, ArcGIS Server, or ArcGIS Enterprise portal for connected workflows or you can create a locator package from it for disconnected workflows.

In either case your app consumes the locator via a `LocatorTask`, as described in Search for places.

You can also use the user's current location as the from- or to- location.

For simplicity, the code below uses two pre-defined stops instead of using a locator/having the user select the stops. These stops are added to the route parameters.

```
// Create map points for locations on the network
var stopPoint1 = new MapPoint(-13041722.4426145, 3857785.86358909,
SpatialReferences.WebMercator);
var stopPoint2 = new MapPoint(-13042811.1283018, 3857081.41990909,
SpatialReferences.WebMercator);

// Create stops
var stop1 = new Stop(stopPoint1);
var stop2 = new Stop(stopPoint2);

// Add stops to a list
List<Stop> stops = new List<Stop>();
stops.Add(stop1);
stops.Add(stop2);

// Set the stops in the route parameters
routeParams.SetStops(stops);
```

3. Solve the route. You do this using the `SolveRouteAsync` method of the route task, passing in the parameters containing your route stops.

```
RouteResult routeResult = await routeTask.SolveRouteAsync(routeParams);
```

4. Do one or both of the following to prepare for displaying the resulting route when it's created:
   - To display the resulting route lines on the map, create a `GraphicsOverlay` and read the geometry of the route from the route result.

```
// get the route from the results
// Verify that at least one route is in the result
if (routeResult != null && routeResult.Routes != null && routeResult.Routes.Count > 0)
{
    // Loop through the route results and draw route geometry as graphics
    foreach (Route r in routeResult.Routes)
    {
        // Create a graphic using the route (polyline) geometry
        Graphic routeGraphic = new Graphic(r.RouteGeometry as Polyline);

        // Add the graphic to a graphics overlay in the map view
        MyMapView.GraphicsOverlays[0].Graphics.Add(routeGraphic);
    }
}
```

   - You can get turn-by-turn directions by listing each `DirectionManeuver` from the route result. Each step of the route is represented in a `DirectionManeuver` and the class contains time, distance, geometry, and direction text describing each

Copyright © 1995-2020 Esri. All rights reserved.

maneuver. The code below shows the extraction of the direction text for each maneuver along the route. The route steps are displayed in a list in the sample application.

```
// Loop through the direction maneuvers for each route and build directions text
StringBuilder directionsBuilder = new StringBuilder();
foreach (DirectionManeuver direction in r.DirectionManeuvers)
{
    directionsBuilder.AppendLine(direction.DirectionText);
}

// Get the ordered set of maneuvers as a string
var directionsText = directionsBuilder.ToString();
```

How much will my routing app cost?

The cost points to consider when creating your routing app are the following:

- You might need a paid deployment plan:
  - If you want to deploy an app that you charge users for and your app uses ArcGIS Online
  - If you want to deploy an app that uses more than 50 ArcGIS Online credits per month
- If you want to deploy an app that uses more ArcGIS Online credits than the number of credits that came with your ArcGIS Developer Subscription, then you'll need to do one of the following:
  - Purchase credits as you use them
  - Purchase a deployment plan
  - Use named users and pass the cost of credits on to those users
- Named users in a production environment have a cost associated with them. This cost is not necessarily your cost, as the developer. Whoever owns the organization on ArcGIS Online or whoever owns the license for ArcGIS Enterprise portal pays for the named users.
- If your app consumes services hosted by ArcGIS Server or ArcGIS Enterprise portal, then there is a cost (not necessarily to you, the developer) for licensing ArcGIS Server or ArcGIS Enterprise portal in order to host those services.
- If your app uses a Basic, Standard, Advanced, or Analysis license level of ArcGIS Runtime, then you must purchase that license. If your app runs offline, you may have to purchase deployment packs. For a list of capabilities and the license level they require, see Licensing capabilities in "License your app."

Copyright © 1995-2020 Esri. All rights reserved.

Navigate a route

ArcGIS Runtime SDK can be used to find a route between two or more locations across a transportation network. The route result can include estimated travel time and distance, as well as driving directions for traversing the route. The navigation API allows you to further enhance the routing experience using the current device location to track progress and provide navigation instructions (maneuvers) as the user travels the route. You can integrate driving directions with your device's text-to-speech capability and automatically recalculate a new route when the user leaves the current one.

The route tracker object provides the following functionality using the current device location and an appropriate route result:

- Progress information relative to the next stop, to the next maneuver, or along the entire route
- Guidance for navigation as it's needed (when approaching a maneuver, for example)
- Automatic recalculation of a route if the device location goes off route

The route result used for navigation must include stops and directions. To enable route recalculation while navigating, you must also provide the route task and routing parameters used to generate the route.

Track progress along a route

Once a route has been calculated that includes at least two stops (a start location and one or more destinations, in other words) and driving directions, you can use the route tracker to track progress as the user traverses the route. To track the current device location, you must listen for location change events on the map view location display. As the device moves, pass the new location to the route tracker to update things such as the estimated remaining travel time and distance.

1. Create a route between specified locations. Generally, the route starts at the user's current location. In the route parameters, make sure to include stops and driving directions in the results.

```
// Get the default route parameters.
RouteParameters routeParams = await routeTask.CreateDefaultParametersAsync();
// Make sure to return stops and directions in the results.
routeParams.ReturnDirections = true;
routeParams.ReturnStops = true;

// Create the route using these parameters ...
```

2. To show progress along a route, display two polylines: one to represent the portion of the route that has been traveled (from the start to the current location) and another to show the portion that remains (from the current location to the destination). Add these geometries to the map view as graphics (using distinct symbols) and update them from the route tracker as the route changes.

```
// Create a graphic to represent the route that remains to be traveled (initially the entire
route).
SimpleLineSymbol routeSymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Dash,
Color.BlueViolet, 5);
_routeAheadGraphic = new Graphic(route.RouteGeometry, routeSymbol);

// Create a graphic to represent the route that's been traveled (initially empty).
routeSymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Solid, Color.LightBlue, 3);
_routeTraveledGraphic = new Graphic { Symbol = routeSymbol };
```

3. Create a route tracker and pass in a route to travel. Handle the tracking status changed event to provide information such as the distance or time remaining to the destination. To update driving instructions when they are needed, handle the new voice guidance event.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Pass the route result to a new route tracker along with the index of the route to navigate.
_tracker = new RouteTracker(routeResult, 0);

// Handle new voice guidance notifications.
_tracker.NewVoiceGuidance += Tracker_NewVoiceGuidance;

// Handle route tracking status changes.
_tracker.TrackingStatusChanged += Tracker_TrackingStatusChanged;
```

4. To enable rerouting, you must verify that the route task supports it. You must then provide the route task and parameters that were used to create the original route.

```
// See if the route task supports rerouting when a user goes off-route.
if (routeTask.RouteTaskInfo.SupportsRerouting)
{
    // If supported, enable rerouting by passing in the route task and parameters, a routing
strategy, and whether new routes should visit first stop.
    await _tracker.EnableReroutingAsync(routeTask, routeParams,
ReroutingStrategy.ToNextWaypoint, false);

    // Handle rerouting completion to display updated route graphic and report new status.
    _tracker.RerouteCompleted += Tracker_RerouteCompleted;
}
```

Note:

Use `RouteTask.RouteTaskInfo.SupportsRerouting` to see if the underlaying route service or dataset supports recalculation of routes.

5. Enable location display on the map view and handle device location updates by passing the new location to the route tracker. The tracker will update the status when it gets a new location, including whether the user is off route.

```
// Handle the location changed event for the map view location display.
MyMapView.LocationDisplay.DataSource.LocationChanged += (sender, location) =>
{
    Dispatcher.BeginInvoke((Action)delegate ()
    {
        // Zoom in on first location received and set the location auto pan mode for
navigation.
        if (MyMapView.LocationDisplay.AutoPanMode != LocationDisplayAutoPanMode.Navigation)
        {
            MyMapView.SetViewpointCenterAsync(location.Position, 5000);
            MyMapView.LocationDisplay.AutoPanMode = LocationDisplayAutoPanMode.Navigation;
        }

        // Use navigation tracker to update navigation status with the current location.
        // (This will fire the RouteTracker.TrackingStatusChanged event).
        _tracker.TrackLocationAsync(location);
    });
};

// Enable location display to start receiving location events.
MyMapView.LocationDisplay.IsEnabled = true;
```

Copyright © 1995-2020 Esri. All rights reserved.

Note:
To test or demonstrate navigation functionality, you can use a simulated location data source to follow points along a predefined path. See Simulate device location updates in the "Show device's location" topic for details. To test successfully navigating a route, for example, you could use the route geometry (polyline) as the path to follow in the simulated location data source.

6. In the tracking status changed event handler, create a status update for the user. This might be UI updates that include the travel time or distance remaining for the route or until the next maneuver (turn, for example). You can also warn the user if they are off route (especially if rerouting is not enabled). See the Report progress section for more information about the types of progress provided by the route tracker.
The following code checks whether the user is off route and either provides a warning or updates the remaining distance and time to the destination. It also updates the graphics that show the portion of the route that's been traveled and the portion that remains.

```
//Function to handle route tracker status changes.
private void Tracker_TrackingStatusChanged(object sender,
RouteTrackerTrackingStatusChangedEventArgs e)
{
// Get the tracking status passed into the handler.
TrackingStatus status = e.TrackingStatus;

// Create a string builder to build a status message.
System.Text.StringBuilder statusMessageBuilder = new System.Text.StringBuilder("Route Status\
n");

// See if the user is on route.
if (status.IsOnRoute)
{
    // Report the distance and time remaining to travel the current route.
    statusMessageBuilder.AppendLine("Distance remaining: " +
                                    status.RouteProgress.RemainingDistance.DisplayText + " "
+

status.RouteProgress.RemainingDistance.DisplayTextUnits.PluralDisplayName);
    statusMessageBuilder.AppendLine("Time remaining: " +

status.RouteProgress.RemainingTime.ToString(@"hh\:mm\:ss"));

    // Get the lines representing the route ahead and the route already traveled.
    Polyline lineToTravel = status.RouteProgress.RemainingGeometry;
    Polyline lineTraveled = status.RouteProgress.TraversedGeometry;

    // Update the route graphics.
    _routeAheadGraphic.Geometry = lineToTravel;
    _routeTraveledGraphic.Geometry = lineTraveled;
}
else
{
    // If off route, warn the user.
    statusMessageBuilder.AppendLine("Off route!");
}

// Get the status message from the string builder.
string statusMessage = statusMessageBuilder.ToString();

// TODO: update the UI with the status message.
// ...
```

Copyright © 1995-2020 Esri. All rights reserved.

7. Use the new voice guidance event handler to give driving instructions to the user. The voice guidance object that's passed into the event contains text that you can pass to a text-to-speech engine or use to update instructions shown in the UI. The following code handles the `NewVoiceGuidance` event to show instructions for the next maneuver.

```
private void Tracker_NewVoiceGuidance(object sender, RouteTrackerNewVoiceGuidanceEventArgs e)
{
    Dispatcher.Invoke(() =>
    {
        ManeuverTextBlock.Text = e.VoiceGuidance.Text;
    });
}
```

8. If route recalculation is enabled, you can handle the route tracker's reroute completed event to update the route (a graphic, for example) displayed in the map view. See the Handle rerouting section for more information.

```
// Function to handle rerouting.
private void Tracker_RerouteCompleted(object sender, RouteTrackerRerouteCompletedEventArgs e)
{
    // Get the geometry (polyline) for the recalculated route.
    Geometry newRoutePolyline = e.TrackingStatus.RouteResult.Routes[0].RouteGeometry;

    // Update the "route ahead" graphic with the new line.
    _routeAheadGraphic.Geometry = newRoutePolyline;
}
```

Report progress

The route tracker's `TrackingStatus` object provides values that you can use to report progress as the user traverses a route. This object is available from the route tracker directly or when the tracking status changes, allowing you to evaluate route progress any time it changes. The basic status it provides is whether the user is still traveling the route. If rerouting is enabled for the tracker, a new route will be created whenever this value is false (if the current location is on the network).

The tracking status object also provides progress relative to the following parts of the route:

- Destination—Describes progress to the next stop in the route
- Maneuver—Describes progress to the next maneuver (driving instruction, for example)
- Route—Describes progress along the entire route

Each of these types of progress is represented by the `TrackingProgress` object with the following information:

- Remaining distance—The distance remaining to the relevant location (next stop, next maneuver, or route end)

Copyright © 1995-2020 Esri. All rights reserved.

- Remaining geometry—A polyline representing the portion of the route between the current location and the relevant location (next stop, next maneuver, or route end)
- Remaining time—The time remaining to the relevant location (next stop, next maneuver, or route end)
- Traversed geometry—A polyline representing the portion of the route between the start of the route and the current location

The tracking status also allows you access to the route that's being traversed. Using the current maneuver index, you can find the current maneuver from the route's list of directions.

```
// Get the tracking status passed into the handler.
TrackingStatus status = e.TrackingStatus;

// Get the route object from the status.
Route thisRoute = status.RouteResult.Routes[0];

// Get the current maneuver.
DirectionManeuver thisManeuver = thisRoute.DirectionManeuvers[status.CurrentManeuverIndex];
```

Voice guidance

To give instructions to the user as they are approaching a maneuver, you can handle the new voice guidance event. Voice guidance notifications are generated at the time they are needed while traveling the route. You can use the text provided by the voice guidance to play the instruction using an available text-to-speech engine, or update driving instructions text in the UI.

The current speed, distance to the next maneuver, and the time required to enunciate an instruction are all considered when generating text for voice guidance. The driving directions text from the route and the distance to the maneuver can be used in the guidance, such as "in 300 meters, turn right on VENICE BLVD". Voice guidance text can be categorized as one of the following types of notifications:

- Long—Occurring immediately after a maneuver starts to describe the next one, it is formatted using the full directions text. "Go straight along Main Street, and in one mile turn right on First Street", for example.
- Moderate—Occurring 20 to 30 seconds before the maneuver, it is formatted using a parsed down form of the direction text. "In half a mile, turn right on First Street", for example.
- Short—Occurring approximately 10 seconds before the maneuver, it is formatted using a terse form of the direction text. "Turn right on First Street", for example.
- Combined—Two consecutive short maneuvers can be combined into a single voice guidance. "Turn right on First Street, then left", for example.

For any particular maneuver, one or more of these types of notifications can be generated. Abbreviations in directions text can be expanded when the voice guidance is created. "W Carolina destination is ahead" can be generated as "West Carolina destination is ahead", for example. Characters that are used it text directions, such as slashes, dashes, and so on, can be dropped from the voice guidance text.

Handle rerouting

You can enable rerouting for a route tracker only if the underlying route data (online service or local dataset) supports rerouting. Currently, rerouting is only supported for routes created from a local network dataset. You can check the route task information to verify that rerouting is supported. If it is, you can enable rerouting on the route tracker. To enable rerouting, you need the following:

- The original route task that was used to calculate the route used by the tracker
- The original route task parameters used to calculate the tracker route

Copyright © 1995-2020 Esri. All rights reserved.

- One of the following rerouting strategy values to use for the creation of new routes
  - Resequence stops only—Optimize the new route for the remaining stops.
  - To next stop—Reroute to the next unvisited stop in the original order.
  - To next waypoint—Reroute to the next unvisited waypoint, rest break, or stop.
- Knowledge of whether the first stop in the route must be visited when creating the new route

Note:
Rerouting occurs when the user is off route and the current location is on the network. If, for example, the current location is in a parking lot that is not part of the network, a new route will not be calculated unless the user approaches the network while still off the original route. This avoids unnecessary rerouting calculations.

You can cancel rerouting while a new route is being calculated and disable rerouting to turn that functionality off. Route tracker events allow you to respond when calculation of a new route begins and when it completes. You can use the completion event to display the new route in the map view.

Copyright © 1995-2020 Esri. All rights reserved.

Route to closest location

You can do the following using the ArcGIS Runtime SDK closest facilities task:

- Find the nearest facilities to incidents based on network travel time
- Set a maximum impedance (distance or time for example) to search for facilities
- Set the number of facilities to find for each incident
- Avoid restricted areas and maneuvers
- Specify different travel modes to model various transportation scenarios

A closest facilities task is a network analysis operation that is executed asynchronously. It returns results with details regarding the closest facilities to incidents and the routes to travel for a given transportation network. This operation is sometimes referred to as "solving for closest facilities" or "finding closest facilities".

Transportation network datasets are modeled from features (lines and points) that represent roads, bridges, tunnels, bike paths, train tracks, junctions, and other elements in the network. The geometric intersections of features help to define connectivity between the network elements they represent. The connectivity of the transportation network is analyzed to find the best routes between incidents and facilities.

Note:
To learn more, see What is a network dataset? in the ArcGIS Desktop documentation.

Choose a data source

The closest facilities task supports using online services or a local mobile geodatabase as its data source for a network.

Use an online network service

To model travel between incidents and facilities, you can use a transportation network exposed as an online ArcGIS service (an ArcGIS network analysis service). Network Analysis services are hosted in Esri's cloud platform, ArcGIS Online, or can be published on your own ArcGIS servers. These services provide a REST API for clients such as mobile and web applications. For more information about publishing a service, see Tutorial: Publishing a network analysis service.

The most important criterion for choosing a service is the area that it covers. You cannot find the closest facilities using a service that does not encompass all the incidents, facilities, and barriers that your user will specify. In addition, services offer differing levels of capability and specialization, which are determined by the service properties and the underlying transportation network dataset.

You can build a transportation network dataset from your own data, or use a dataset provided by someone else, such as Esri's World Street Map data. Then, you can publish the data as a closest facility service on your own portal or on ArcGIS Online. Or, you could use a closest facility service created by someone else that suits your needs, such as the Esri World Closest Facility Service, which references data that covers much of the world. Since the ArcGIS Online data also includes traffic for many areas, analysis can account for dynamic traffic conditions. For instance, you can find the closest facility for a departure time of 7:00 a.m. and compare that route with one you generate for 8:00 a.m.

Accessing a closest facility service may require credentials, which is also true for accessing map and feature services. You provide these credentials when you create the closest facilities task object or by adding a token to the `AuthenticationManager`. For more information, see Use the Authentication Manager.

Copyright © 1995-2020 Esri. All rights reserved.

When using ArcGIS Enterprise portal, default utility services may be configured, such as a closest facilities service. ArcGIS Runtime API provides methods to obtain the default services' URLs for a given portal. Setting a default service on a portal is optional, however, so they may not always be set.

Note:
A closest facilities geoprocessing service can also be published and consumed. For more information on publishing, see the Find Closest Facilities tool and the Publishing a geoprocessing service help topics. For consuming geoprocessing services from Runtime, see Geoprocessing.

Use local data

You can find closest facilities using a transportation network stored on your device, meaning that your device does not need a network connection to a communication network to find a route. You have a few options for provisioning your device with a local transportation network: mobile geodatabases, mobile map packages or mobile scene packages.

If you'd like a ready-to-use and regularly updated network dataset (and locator) for your area of interest, you can license StreetMap Premium data (in mobile map package format). For details, see Add StreetMap Premium data.

To learn more about mobile geodatabases see Creating ArcGIS Runtime content.

To learn more about mobile packages see Create a mobile map packagewith ArcGIS Pro, or Create a mobile scene package with ArcGIS Pro.

Find the closest facilities

The basic steps for finding the closest facilities are outlined below.

1. Create and load the closest facilities task
2. Specify closest facility task parameters
3. Execute the closest facilities task
4. Process results

Create and load the closest facilities task

The closest facilities task is an object that references a closest facilities service or local transportation network and accepts parameters that define how closest facilities should be found on the network. When executed, it returns routes to the closest facilities as results.

The following example creates a closest facilities task from a public service:

```
var uri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/NetworkAnalysis/
SanDiego/NAServer/ClosestFacility");
ClosestFacilityTask closestFacilityTask = await ClosestFacilityTask.CreateAsync(uri);
```

You can create a closest facilities task from a portal's default service (if one is defined) as follows:

```
// connect to the specified portal
var portalUri = new Uri("http://" + portalHost + "/sharing/rest");
ArcGISPortal portal = await ArcGISPortal.CreateAsync(portalUri);

// generate an authentication token for the portal
```

Copyright © 1995-2020 Esri. All rights reserved.

```
TokenCredential token = await AuthenticationManager.Current.GenerateCredentialAsync(portalUri,
userName, password);

// get the Uri for the default closest facility service
var taskUri = new Uri(portal.PortalInfo.HelperServices.ClosestFacilityService.Url);
// create a closest facility task from the Uri
ClosestFacilityTask task = await ClosestFacilityTask.CreateAsync(taskUri, token);
```

You can also create a closest facilities task from a mobile map package or mobile geodatabase stored on the device.

```
// open a local mobile map package (.mmpk file)
MobileMapPackage mmpk = await MobileMapPackage.OpenAsync(mapPackagePath);

// load the first map in the package
Map firstMap = mmpk.Maps[0];
await firstMap.LoadAsync();

// create a closest facility task from the first transportation network in the map
ClosestFacilityTask task = await
ClosestFacilityTask.CreateAsync(firstMap.TransportationNetworks[0]);
```

```
// create a closest facility task from a mobile geodatabase (.geodatabase) and network name
ClosestFacilityTask task = await ClosestFacilityTask.CreateAsync(geodatabasePath, "Streets_ND");
```

Specify closest facility task parameters

Closest facility task parameters specify how the analysis should be performed. A variety of parameters are available that let you customize how the closest facilities are determined, such as incident locations, facility locations, barriers to network travel, travel mode, target facility count, output spatial reference, and so on.

You can get the default set of closest facility task parameters from the closest facility task object. These default parameters are determined by properties of the service and the underlying transportation network dataset. Different services will most likely have different defaults. Rather than building task parameters from scratch, it's recommended to start with the default parameters for the service and to then modify individual parameter values you want to change.

The following example gets the default `ClosestFacilityParameters` and changes some of the parameter values:

```
// get the default parameters defined for the underlying service
ClosestFacilityParameters parameters = await closestFacilityTask.CreateDefaultParametersAsync();

// explicitly set values for some parameters
parameters.ReturnDirections = true;
parameters.ReturnRoutes = true;
parameters.OutputSpatialReference = myMapView.SpatialReference;
```

Your app might allow users to choose some parameters directly, such as cost attributes to accumulate or the number of facilities to find. Your app would get these choices from the user and set the corresponding values on the parameter object.

Copyright © 1995-2020 Esri. All rights reserved.

Specify incidents, facilities, and barriers

Incidents and facilities are points features. They can be created either explicitly from `Point` objects or by specifying a feature table (local or online) with a geometry of type point. When using a feature table, you can optionally set a where clause to filter which features to use. When specifying an online feature table, the table will not be queried until solve is called.

Properties on `Facility` and `Incident` classes can be used to model, and to accurately locate, moving features used as these inputs. You don't need to provide values for these properties if using stationary facilities or incidents.

- **Speed**—how fast the input is moving (meters per second). Speed is usually obtained from the GPS receiver and updated as the location changes
- **Bearing**—the direction in which a point is moving (degrees measured clockwise from true north). Bearing is usually obtained from the GPS receiver and updated as the location changes
- **Bearing tolerance**—a range of acceptable bearing values when locating moving points on an edge, using the specified bearing
- **Latency**—indicates how much time is expected to elapse from the moment GPS information is sent from a moving vehicle to a server and the moment the processed route is received by the vehicle's navigation device

Barriers are locations along the network that either restrict travel completely (a road closure for example), or add to the cost of travel when traversed (road work or increased traffic for example).

Barriers can be points, polylines, or polygons, and you can define barriers in various ways. Your users might draw barrier graphics interactively, your app might load graphics from a feature table, and so on. For example, you may have a feature table with areas affected by a forest fire that must be avoided. Barriers are optional when solving for closest facilities, but when relevant, can make your results more accurate.

The solve operation projects input features to the spatial reference of the transportation network dataset used by the service. The spatial reference of the incidents, facilities, and barriers must be defined but can be different than the transportation network dataset's spatial reference.

The following example creates incidents, facilities, and barriers objects and uses them to set properties on the closest facility parameters:

```
// create a incident from a location
var incidentOneLocation = new MapPoint(-13034379.71, 3858390.14, SpatialReferences.WebMercator);
var incidentOne = new Incident(incidentOneLocation);

// create a list of incidents and add to it
var incidentList = new List<Incident>();
incidentList.Add(incidentOne);

// assign the incident list to the parameters
parameters.SetIncidents(incidentList);

// create a facility from a location
var facilityOneLocation = new MapPoint(-13032763.24, 3860880.60, SpatialReferences.WebMercator);
var facilityOne = new Facility(facilityOneLocation);

// create a list of facilities and add to it
var facilityList = new List<Facility>();
facilityList.Add(facilityOne);

// assign the facilities list to the parameters
parameters.SetFacilities(facilityList);

// create a PointBarrier to avoid
```

Copyright © 1995-2020 Esri. All rights reserved.

```
var accidentLocation = new MapPoint(-13033331.71, 3859368.54, SpatialReferences.WebMercator);
var barrier = new PointBarrier(accidentLocation);

// create a list of point barriers and add to it
var pointBarriers = new List<PointBarrier>();
pointBarriers.Add(barrier);

// assign the point barriers list to the parameters
parameters.SetPointBarriers(pointBarriers);
```

You can also use online tables to define facilities and incidents as follows:

```
// open a service table of facilities
Uri facilityTableUri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/
NetworkAnalysis/SanDiegoInputs/MapServer/0");
ServiceFeatureTable facilityServiceTable = new ServiceFeatureTable(facilityTableUri);

// create query parameters to select all facilities
QueryParameters facilityQueryParams = new QueryParameters();
facilityQueryParams.WhereClause = "1=1";

// set the task parameters facilities using the table and the query parameters
closestFacilityparameters.SetFacilities(facilityServiceTable, facilityQueryParams);

// open a service table of incidents
Uri incidentTableUri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/
NetworkAnalysis/SanDiegoInputs/MapServer/1");
ServiceFeatureTable incidentServiceTable = new ServiceFeatureTable(incidentTableUri);

// create query parameters to select all incidents
QueryParameters incidentQueryParams = new QueryParameters();
incidentQueryParams.WhereClause = "1=1";

// set the task parameters incidents using the table and the query parameters
closestFacilityparameters.SetIncidents(incidentServiceTable, incidentQueryParams);
```

The following example uses a local table to define facilities:

```
// open a local runtime geodatabase with tables for task inputs
var geodatabasePath = @"c:\data\mydata.geodatabase";
var gdb = await Geodatabase.OpenAsync(geodatabasePath);

// get the table that contains facility features
ArcGISFeatureTable facilityTable = gdb.GeodatabaseFeatureTable("Facilities");

// create a query parameters that filters for hospitals and fire stations
var facilityQueryParams = new QueryParameters();
facilityQueryParams.WhereClause = "Type='Hospital' or Type='Fire Station'";

// set the facilities using the local table and query
closestFacilityparameters.SetFacilities(facilityTable, facilityQueryParams);

// get the table that contains incident features
ArcGISFeatureTable incidentTable = gdb.GeodatabaseFeatureTable("Incidents");

// create a query parameters that gets all incidents
var incidentsQueryParams = new QueryParameters();
incidentsQueryParams.WhereClause = "1=1";
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// set the incidents using the local table and query
closestFacilityparameters.SetIncidents(incidentTable, incidentsQueryParams);
```

Note:

The input table's field values can be used to populate properties for incidents, facilities, and barriers. For local geodatabases, this is based on a naming convention. For details, see Closest facility analysis. When publishing a closest facility service, the default names can be altered. For details, see Loading network analysis objects in ArcMap.

If your app shows the device's location, you may wish to use the geometry of this location for the incident geometry.

Other closest facility parameters

Additional properties on the closest facility parameters object allow you to refine how travel across the network is modeled, to place thresholds on the solve, and to restrict the results that are returned. The parameters are as follows:

- **Accumulate attribute names**—A list of cost attributes to accumulate. Cost attributes for a particular service can be obtained from `ClosestFacilityTaskInfo.CostAttributes`.
- **Travel mode**—The type of travel used when solving for closest facilities. Travel modes on a network dataset define how a pedestrian, car, truck, or other medium of transportation moves through the network. A list of travel modes for a particular service can be returned using `ClosestFacilityTaskInfo.TravelModes`.
- **Default target facility count**—The maximum number of facilities to be found for each incident. This value must be at least 1 or more. Individual incidents can override the default value by specifying **target facility count**.
- **Default impedance cutoff**—An impedance value after which the solve will stop searching for facilities. Any facilities beyond this value will not be include in the results.
- **Start time**—The date and time used for solving for closest facilities. This can affect the solve result if the network data source incorporates historic or live traffic.
- **Start time usage**—Indicates whether the start time represents the departure or arrival along the closest facility routes.
- **Travel direction**—Indicates whether the direction of travel is from incidents to facilities or from facilities to incidents. On a network with one-way restrictions and traffic data, changing the travel direction can produce different results. The direction you choose depends on the nature of your analysis.

Execute the closest facilities task

To find closest facilities, call the asynchronous `SolveClosestFacilityAsync` method and await the result. A task parameters object (`ClosestFacilityParameters`) that defines input features and task preferences is passed to the method, and a `ClosestFacilityResult` object is returned.

```
// solve for closest facilities
ClosestFacilityResult solveResult = await
closestFacilityTask.SolveClosestFacilityAsync(_closestFacilityParameters);

// ... process results ...
```

Copyright © 1995-2020 Esri. All rights reserved.

Process results

When the closest facility task completes, results are available as a `ClosestFacilityResult` object. Depending on the success of the task, the `Messages` property on the `ClosestFacilityResult` object may be populated. These messages contain information about how the task was solved.

Output incidents, facilities, or barriers

If the task parameters are set to return incidents, facilities, or barriers, properties of the closest facility result for these outputs are populated. These output features are copies of the corresponding inputs with additional properties such as `LocationStatus` and `DistanceToNetworkLocation`. The location of output incidents and facilities may also differ slightly from the input locations since output incidents and facilities are snapped to the closest non-restricted network element.

Incident to facility routes

Depending on the input parameters, an incident may be routed to multiple facilities. To retrieve each of these routes from the results, use the corresponding facility and incident indices. To help with this, `GetRankedFacilities` returns the output facilities indices in order. The following example shows how to retrieve the routes in order:

```
// Iterate over the solve result incidents
for (int incidentIndex = 0; incidentIndex < solveResult.Incidents.Count; incidentIndex++)
{
    // get the rankings of the facilities for this incident (an ordered list of the returned
facilities indexes)
    // --Example: if the facility ranking is [2,0,1,3] then ClosestFacilityResult.Facilities[2]
is the closest, followed
    //            by ClosestFacilityResult.Facilities[0], and so on.
    IReadOnlyList<int> rankedFacilitiesIndexes =
solveResult.GetRankedFacilityIndexes(incidentIndex);
          // using the ranked list of facilities, get the route for this incident-facility
```

Copyright © 1995-2020 Esri. All rights reserved.

```
combination
    // the facility indexes are in ranked order, so the 'best' route will be first, followed by
the second best, and so on
    foreach (var facilityIndex in rankedFacilitiesIndexes)
    {
        ClosestFacilityRoute closestFacilityRoute = solveResult.GetRoute(facilityIndex,
incidentIndex);
        // ... work with the closestFacilityRoute ...
    }
}
```

Each closest facility route in the results contains the polyline between an incident and facility, as well as the time spent traveling the route and the total time (travel time plus wait time if any).

Directions

Directions are returned for each route in the results if requested in the task parameters. Depending on the service configuration, directions can be returned in more than one language. Supported languages for a service are available in the closest facility task info.

Directions are composed of direction maneuvers. Each maneuver contains properties such as the direction text, which contains instructions that describe travel through a section of a route. A direction maneuver is further broken down into maneuver messages that give details about a specific maneuver.

Driving Directions

1: Start at Incident A
2: Go south on 6TH AVE toward BEECH ST
3: Turn right on BEECH ST
4: Turn right on 5TH AVE
5: Turn right onto ramp and go on I-5 S
6: Bear right onto ramp to PERSHING DR toward B STREET
7: At fork keep left on PERSHING DR
8: Continue on PERSHING DR
9: Turn left on FLORIDA DR
10: Turn right on MORLEY FIELD DR
11: Continue on ALABAMA ST
12: Finish at Facility X

If defined in the transportation network, directions may also include direction events, which are notifications that do not require any action when traversing the route, for example, crossing over a time zone boundary.

Direction maneuver's can include **from level** and **to level** values to define things like floor levels inside a building. For example, a user may want to see just one level of a walking route that spans multiple floors.

In order to return from level and to level values for a route:

- The route's directions style must be `Campus`
- The network dataset must be configured to support levels as described in the Setting directions topic in the desktop help. This applies to both local data and the underlying network used by a routing service
- The ArcGIS Server version hosting the service must be 10.6 or higher (when using online routing services)

Copyright © 1995-2020 Esri. All rights reserved.

Generate service areas

You can do the following using the ArcGIS Runtime SDK service area task:

- Find all roads that can be reached within a given distance or time from a specific location
- Request multiple service areas cutoffs for one or more locations
- Specify how service areas from multiple locations are constructed
- Use different travel modes to model various transportation scenarios

This topic describes how to generate service areas from an online network service. For an overview of networks and network analysis, see What is a network dataset? in the ArcGIS Desktop documentation.

Choosing a data source

The service area task supports using online services or a local mobile geodatabase as its data source for a network.

Use an online network service

To model service area boundaries for a set of facilities, you can use a transportation network exposed as an online ArcGIS service (an ArcGIS network analysis service). Network analysis services are hosted in Esri's cloud platform, ArcGIS Online, or can be published on your own ArcGIS servers. These services provide a REST API for clients such as mobile and web applications. For more information about publishing a service, see Tutorial: Publishing a network analysis service.

The most important criterion for choosing a service is the geographic extent that it covers. You cannot find service areas using a service that does not encompass the street network used by the facilities. In addition, services offer differing levels of capability and specialization, which are determined by the service properties and the underlying transportation network dataset.

You can build a transportation network dataset from your own data, or use a dataset provided by someone else, such as Esri's World Street Map data. Then, you can publish the data as a service area service on your own portal or on ArcGIS Online. Or, you can use a service area service created by someone else that suits your needs, such as the Esri World Service Area Service, which references data that covers much of the world. Since the ArcGIS Online data also includes traffic for many areas, analysis can account for dynamic traffic conditions. For instance, you can generate service areas for 7:00 a.m. and compare them with those you generate for 8:00 a.m.

Accessing a service area service may require authorization, which is also true for accessing map and feature services. You can authorize by providing valid credentials when you create the service area task object or by adding a token to the `AuthenticationManager`. For more information, see Use the Authentication Manager.

When using ArcGIS Enterprise portal, default utility services may be configured, such as a service area service. ArcGIS Runtime API provides methods to obtain the default service's URL for a given portal. Setting a default service on a portal is optional, however, so it may not always be set.

Note:
A service area geoprocessing service can also be published and consumed. For more information on publishing, see the Generate Service Areas topic and Publishing a geoprocessing service. For more information about consuming geoprocessing services from Runtime, see Geoprocessing.

Copyright © 1995-2020 Esri. All rights reserved.

Use local data

You can find service areas using a transportation network stored on your device, meaning that your device does not need a network connection to a communication network to find a service are. You have a few of options for provisioning your device with a local transportation network: mobile geodatabases, mobile map packages or mobile scene packages..

If you'd like a ready-to-use and regularly updated network dataset (and locator) for your area of interest, you can license StreetMap Premium data (in mobile map package format). For details, see Add StreetMap Premium data.

To learn more about mobile geodatabases see Creating ArcGIS Runtime content.

To learn more about mobile packages see Create a mobile map package or Create a mobile scene package with ArcGIS Pro.

Generate service areas

The basic steps for finding service areas are outlined below.

1. Create and load service area task
2. Specify service area task parameters
3. Execute the service area task
4. Process results

Create and load the service area task

The service area task is an object that refers to a service or local transportation network, accepts service area task parameters, is executed to find the service area, and returns results.

The following creates a service area task from a public service:

```
Uri uri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/NetworkAnalysis/
SanDiego/NAServer/ServiceArea");
ServiceAreaTask serviceAreaTask = await ServiceAreaTask.CreateAsync(uri);
```

The following creates a service area task from a portal's default service:

```
var portalUri = new Uri("http://" + portalHost + "/sharing/rest");
ArcGISPortal portal = await ArcGISPortal.CreateAsync(portalUri);

// generate an authentication token for the portal
TokenCredential token = await AuthenticationManager.Current.GenerateCredentialAsync(portalUri,
"<user name here>", "<password here>");

// get the Uri for the default service area service
var url = new Uri(portal.PortalInfo.HelperServices.ServiceAreaService.Url);
// create a service area task from the Uri
ServiceAreaTask serviceAreaTask = await ServiceAreaTask.CreateAsync(url, token);
```

You can also create a service area task from a mobile map package or mobile geodatabase stored on the device.

```
// open a local mobile map package (.mmpk file)
MobileMapPackage mmpk = await MobileMapPackage.OpenAsync(mapPackagePath);

// load the first map in the package
Map firstMap = mmpk.Maps[0];
```

Copyright © 1995-2020 Esri. All rights reserved.

```
await firstMap.LoadAsync();

// create a service area task from the first transportation network in the map
ServiceAreaTask task = await ServiceAreaTask.CreateAsync(firstMap.TransportationNetworks[0]);
```

```
// create a service area task from a mobile geodatabase (.geodatabase) and network name
ServiceAreaTask task = await ServiceAreaTask.CreateAsync(geodatabasePath, "Streets_ND");
```

Specify service area task parameters

Service area task parameters specify how the analysis should be performed. There are many parameters that let you customize how the service area is determined. Parameters include facilities, barriers, travel mode, impedance cutoffs, polygon detail, and so on.

You can get the default set of service area task parameters from the service area task object. These default parameters are determined by properties of the service and the underlying transportation network dataset. Often, different services have different defaults. Rather than building task parameters from scratch, it's recommended to start with the default parameters for the service and to then modify the individual parameter values you want to change.

The following example gets the default `ServiceAreaParameters` and changes some of the parameter values.

```
// get the default parameters
ServiceAreaParameters parameters = await serviceAreaTask.CreateDefaultParametersAsync();

// explicitly set values for some parameters
parameters.ReturnPolygons = true;
parameters.PolygonDetail = ServiceAreaPolygonDetail.Standard;
```

Your app might allow users to choose some parameters, such as cost attributes to accumulate or the polygon buffer distance. Your app would get these choices from the user and set the corresponding values on the parameter object.

Specify facilities and barriers

Facilities are points features. They can be created either explicitly from `MapPoint` objects or by specifying a feature table (local or online) with a geometry of type point. When using a feature table you can optionally set a where clause to filter which features to use. When specifying an online feature table, the table will not be queried until solve is called.

Properties on `Facility` can be used to model, and to accurately locate, moving features used as these inputs. You don't need to provide values for these properties if using stationary facilities.

- **Speed**—how fast the input is moving (meters per second). Speed is usually obtained from the GPS receiver and updated as the location changes
- **Bearing**—the direction in which a point is moving (degrees measured clockwise from true north). Bearing is usually obtained from the GPS receiver and updated as the location changes
- **Bearing tolerance**—a range of acceptable bearing values when locating moving points on an edge using the specified bearing
- **Latency**—indicates how much time is expected to elapse from the moment GPS information is sent from a moving vehicle to a server and the moment the processed route is received by the vehicle's navigation device

Copyright © 1995-2020 Esri. All rights reserved.

Barriers are locations along the network that either restrict travel completely (a road closure for example) or add to the cost of travel when traversed (road work or increased traffic for example).

Barriers can be points, polylines, or polygons, and you can define barriers in various ways. Your users might draw barrier graphics interactively, your app might load graphics from a feature table, and so on. For example, you may have a feature table with areas affected by a forest fire that must be avoided. Barriers are optional when solving for service areas but when relevant, can make your results more accurate.

The solve operation projects input features to the spatial reference of the transportation network dataset used by the service. The spatial reference of the facilities and barriers must be defined but can be different than the transportation network dataset's spatial reference.

The following example creates facilities and barriers and uses them to set properties on the service area parameters:

```
// create a facility from a location
var hospitalOneLoc = new MapPoint(-13032763.24, 3860880.60, SpatialReferences.WebMercator);
var hospitalOne = new ServiceAreaFacility(hospitalOneLoc);

// assign the facilities to the parameters
var hospitals = new List<ServiceAreaFacility>();
hospitals.Add(hospitalOne);
parameters.SetFacilities(hospitals);

// create a PointBarrier to avoid
MapPoint accidentLocation = new MapPoint(-13033331.71, 3859368.54,
SpatialReferences.WebMercator);
PointBarrier barrierOne = new PointBarrier(accidentLocation);

// add the point barriers to the route parameters
List<PointBarrier> pointBarriers = new List<PointBarrier>();
pointBarriers.Add(barrierOne);
parameters.SetPointBarriers(pointBarriers);
```

You can also use online tables to define facilities as follows:

```
// open a service table of facilities
Uri facilityTableUri = new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/
NetworkAnalysis/SanDiegoInputs/MapServer/3");
ServiceFeatureTable facilityServiceTable = new ServiceFeatureTable(facilityTableUri);

// create query parameters to select all facilities
QueryParameters facilityQueryParams = new QueryParameters();
```

Copyright © 1995-2020 Esri. All rights reserved.

```
facilityQueryParams.WhereClause = "1=1";

// set the task parameters facilities using the table and the query parameters
parameters.SetFacilities(facilityServiceTable, facilityQueryParams);
```

The following example uses a local table to define facilities:

```
// open a local runtime geodatabase with tables for task inputs
var geodatabasePath = @"c:\data\mydata.geodatabase";
var gdb = await Geodatabase.OpenAsync(geodatabasePath);

// get the table that contains facility features
ArcGISFeatureTable facilityTable = gdb.GeodatabaseFeatureTable("Facilities");

// create a query parameters that filters for hospitals and fire stations
var facilityQueryParams = new QueryParameters();
facilityQueryParams.WhereClause = "Type='Hospital' or Type='Fire Station'";

// set the parameters facilities using the local table and query
parameters.SetFacilities(facilityTable, facilityQueryParams);
```

Note:
The input table's field values can be used to populate properties for facilities and barriers. For local geodatabases, this is based on a naming convention. For details, see Service area analysis. When publishing a service area service, the default names can be altered. For details, see Loading network analysis objects in ArcMap

Other service area parameters properties

The following are the properties that can be set on the parameter's object to further refine how service areas are determined:

- `Default impedance cutoffs`—This property specifies the values used to calculate the extent of the service areas. The inputs are in the units of the impedance cost attribute defined for the network travel mode being used.
  The default impedance cutoffs can be overridden on specific facilities by specifying the impedance cutoffs on a facility. The exception to this is when the **geometry at overlap** parameter is set to **dissolve**, in which case, the facilities cutoffs are ignored and the default impedance cutoff is used. This is to ensure all facilities have the same cutoff values, and the service areas polygons can be dissolved. This behavior is applied even if only a single facility is specified and service area overlap geometry dissolve is specified.
- `Polygon buffer distance`—This property specifies the distance, in meters, by which the outer service area polygons will be trimmed. This is useful if the network is sparse and you don't want the service area to cover large areas where there are no network features.
- `Service area polygon detail`—The level of detail of the output polygon geometry can be specified by setting this property. The higher the requested detail, the more accurate the output but the longer it will take to generate. The following list describes the available values for this property.
  - Generalized
    - Generated quickly
    - Fairly accurate (deteriorates near service area polygon boarders)
    - Polygons are simplified
    - May result in islands of unreached elements being included in the service area

Copyright © 1995-2020 Esri. All rights reserved.

    - Hierarchical solve used
    - Polygons cannot be trimmed
    - Polylines cannot be returned
  - Standard
    - Generated fairly quickly
    - Polygons are simplified
    - Hierarchical solve not used
    - Polygons can be trimmed using the **polygon buffer distance** parameter
    - Polylines can be returned
  - High
    - Longer to generate
    - More accurate
    - Polygons are more detailed
    - Hierarchical solve not used
    - Polygons can be trimmed using the **polygon buffer distance** parameter
    - Polylines can be returned
- `Geometry at overlap`—This property controls how the overlapping polygons are constructed when multiple facilities are specified. The three options are as follows:
  - **Dissolve**—Overlapping polygons with the same cutoff values are merged
  - **Overlap**—Polygons are created for each facility independently and may overlap
  - **Split**—Overlaps between polygons are assigned to a single facility based on the lowest impedance
- `Geometry at cutoffs`—Controls how the output polygons are constructed. If **geometry at cutoffs** is set to **Rings**, the output polygons will not include the area of smaller inner cutoffs. If **geometry at cutoffs** is set to **Disks**, the output polygons will include the area of smaller inner cutoffs.
- `Return polylines`—Lines will only be returned when **polygon detail** is set to **Standard** or **High**. The polylines will be split at the cutoff boundaries.

Execute the service area task

Execute the service area task's solve method, providing the service area parameters.

```
// Solve for service areas
ServiceAreaResult solveResult = await task.SolveServiceAreaAsync(parameters);
// ... process results ...
```

Process results

When the service area task completes, results are available as a service area result. Depending on the success of the task, the `Messages` property on the result object may be populated. These messages contain information about how the task was solved.

Copyright © 1995-2020 Esri. All rights reserved.

The outputs can include polygons, polylines, facilities, and barriers depending on the service area parameter settings. The output facilities and barriers are copies of the corresponding inputs and have information such as location status or distance to network location populated by the solve process.

Service area polygons

To access the resulting service area polygons, you can iterate through the returned facilities and get a list of service area polygons for each facility. The shape and size of these polygons are affected by the service area parameter values passed in to the solve operation.

```
// loop through all facilities in the result
int facilityIndex = 0;
foreach (ServiceAreaFacility f in solveResult.Facilities)
{
    // get the result polygons (service areas) for this facility
    IEnumerable<ServiceAreaPolygon> serviceAreas = solveResult.GetResultPolygons(facilityIndex);
    if (serviceAreas != null)
    {
        // loop through all service areas for this facility
        foreach (var area in serviceAreas)
        {
            // create a fill symbol
            var symbol = new SimpleFillSymbol(SimpleFillSymbolStyle.Solid, GetColor(), null);
            // add a new graphic to display the service area (in a graphics overlay)
            resultOverlay.Graphics.Add(new Graphic(area.Geometry, symbol));
        }
    }

    facilityIndex++;
}
```

Service area polylines

To access the polylines, iterate through the returned facilities and retrieve the list of the network polylines that are within each facility's service areas. The cumulative from and to costs can be retrieved from each polyline.

Copyright © 1995-2020 Esri. All rights reserved.

```
// loop through all facilities in the result
for (int i = 0; i < solveResult.Facilities.Count; i++)
{
    // loop through the polyline results for this facility
    foreach (ServiceAreaPolyline serviceAreaPolyline in solveResult.GetResultPolylines(i))
    {
        // get the cost of traveling this line using the travel mode
        double toCost =
serviceAreaPolyline.GetToCumulativeCost(parameters.TravelMode.ImpedanceAttributeName);
        // ... work with polyline results, summarize costs, and so on ...
    }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Add StreetMap Premium data

StreetMap Premium for ArcGIS Runtime provides enriched street data, which powers a high-quality cartographic map and high-quality search, geocoding, and route analysis. StreetMap Premium maps are consistent across all regions of the world and can be taken offline for disconnected use; they can simultaneously fulfill the need for an address locator, street network dataset, and basemap in your app.

StreetMap Premium delivers data from HERE Technologies as a mobile map package (an `.mmpk` file) for your app to access locally. This MMPK format allows the data to be accessed offline (without a network connection, in other words) and therefore doesn't consume data from your user's data plan. This is the same high-quality data used by ArcGIS Online services, including the World Geocoding Service, Routing Service, and Street Map Service. Instead of spending your time putting together such datasets yourself, you can focus on developing apps that provide advanced searching, geocoding, and routing analysis offline.

StreetMap Premium data is organized into regions that are licensed as extensions to ArcGIS Runtime and are downloaded individually (North America, Latin America, Europe, the Middle East, Africa, and Asia Pacific), allowing your apps to provide a consistent user experience across the globe. Within these regions, maps are available at the sub-region, country, or state/province level. You can even use ArcGIS Pro to clip the data to a custom area of interest.

Note:
StreetMap Premium for ArcGIS Runtime is updated regularly, for example: North America three times per year, Europe two times per year, and Latin America, Asia Pacific, Middle East and Africa once per year.

Follow these general steps to use StreetMap Premium in your ArcGIS Runtime app.

1. Download the StreetMap Premium Greater Los Angeles mobile map package that is provided for development and testing. When you're ready to deploy your app, you'll need to download the required StreetMap Premium packages from My Esri and license StreetMap Premium for each extension (region) your app uses.

   License:
   You do not need a license to develop and test with the Greater Los Angeles mobile map package StreetMap Premium data. When you deploy an app using StreetMap Premium data, however, you will need to license your app as described in the License StreetMap Premium section of this topic. Therefore, while developing and testing, do not include code to license StreetMap. Please contact Esri Customer Service for access to a region of your choice for development and testing or to licence StreetMap Premium data for deployment.

2. Provide the data (mobile map package) for your app. You can provide package(s) with your app or allow the user to download them as needed. Once the package (`*.mmpk`) is available on the client, you can open it to retrieve data, maps, and locators. Using the contents of the package, you can:
   - Display StreetMap Premium data in your app.
   - Locate addresses and places using a StreetMap Premium locator task.
   - Solve routes using the transportation network dataset provided in the StreetMap Premium map package.
3. Attribute StreetMap Premium data somewhere in the app user interface using the words `mapping data from HERE`. If you're attributing more than one data provider, the HERE attribution cannot be less prominent than the attribution for the other data providers. See Attribution in your app for more information.

Copyright © 1995-2020 Esri. All rights reserved.

License StreetMap Premium

Each StreetMap Premium region is licensed as an extension to ArcGIS Runtime. A StreetMap Premium extension license works with all license levels of ArcGIS Runtime (Lite, Basic, Standard, and Advanced). Unlike other ArcGIS Runtime extension licenses, this license does not unlock API capabilities, but rather licenses the use of StreetMap Premium data within one of the available regions. For each region you license, you receive a license key as a "string" to use in your app. These licenses are good for one year, so you must provide a mechanism to notify your users and update the license key for your app when (or before) the license expires.

The following code licenses ArcGIS Runtime and several StreetMap Premium extensions when the app initializes. To update these license keys when they expire, you will need to update and recompile the app code.

```
// ArcGIS Runtime license key
var myRuntimeLicenseKey = "runtimelite,1000,rudxxxxxxxxx,28-feb-2018,xxxxxxxxxxxxxxxxxxxx";

// Extension license keys for various StreetMap Premium areas
var smpNorthAmerica = "runtimesmpna,1000,rudxxxxxxxxx,13-mar-2018,xxxxxxxxxxxxxxxxxxxx";
var smpLatinAmerica = "runtimesmpla,1000,rudxxxxxxxxx,13-mar-2018,xxxxxxxxxxxxxxxxxxxx";
var smpEurope = "runtimesmpe,1000,rudxxxxxxxxx,16-dec-2017,xxxxxxxxxxxxxxxxxxxx";

// Add StreetMap Premium license keys to an array
string[] extensions = { smpNorthAmerica, smpLatinAmerica, smpEurope };

// Set the license for ArcGIS Runtime and the three extensions (areas)
ArcGISRuntimeEnvironment.SetLicense(myRuntimeLicenseKey, extensions);

// Initialize the ArcGIS Runtime before any components are created
ArcGISRuntimeEnvironment.Initialize();
```

You could also read license keys on startup from a text file included with the app and set the licenses. This would allow the user to update license keys in a separate file and would eliminate the need for you to update and recompile the app code.

```
// ... code here to read license keys from a text file into an array of strings ...

// Variable to store the (one) ArcGIS Runtime license key
var runtimeLicense = string.Empty;

// Variable to store the extension license keys (all others in the file)
var extensionLicenses = new List<string>();

// Loop through all license keys from the file
foreach(var s in licenseStrings)
{
    // See if it's an ArcGIS Runtime license key
    if(s.StartsWith("runtimelite") ||
        s.StartsWith("runtimebasic") ||
        s.StartsWith("runtimestandard") ||
        s.StartsWith("runtimeadvanced"))
    {
        // Store the key
        runtimeLicense = s;
    }
    else
    {
        // All other keys are added to the list of extension licenses
        extensionLicenses.Add(s);
    }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Set the licenses before using any ArcGIS Runtime objects
ArcGISRuntimeEnvironment.SetLicense(runtimeLicense, extensionLicenses.ToArray());
```

As the licenses in your app near expiration, you might want to notify the user that new licensing information will be required soon. The following code loops through all extension licenses for the app and notifies the user if a license is within 10 days of expiring.

```
// Get the licensing info for the current app
License licenseInfo = ArcGISRuntimeEnvironment.GetLicense();

// Get all extension licenses
IEnumerable<ExtensionLicense> extLicenses = licenseInfo.Extensions;

// Loop through extension licenses and see when each expires
foreach (ExtensionLicense ext in extLicenses)
{
    // Get license name ("runtimesmpna", e.g.) and expiration date
    var licenseName = ext.Name;
    DateTimeOffset expires = ext.Expiry;

    // Get the number of days until the license expires
    var daysUntilExpired = expires.Subtract(DateTimeOffset.Now).Days;

    // If within 10 days of expiration, notify the user
    if(daysUntilExpired <= 10)
    {
        // ... warn that <licenseName> will expire in <daysUntilExpired> days ...
    }
}
```

License:
If you're building an app for a third-party (if you're the development organization delivering an app to an end-user organization), then that third party must purchase the license directly from Esri and provide the license key(s) to you. For details, see the terms of use.

For details on licensing your ArcGIS Runtime app, see License your app.

Display StreetMap Premium data

Inside each StreetMap Premium mobile map package, you'll find two maps: **Navigation Day** and **StreetMap Day**. Each of these maps display the same data and use similar symbology for the layers. They also use scale dependent rendering to improve display performance and readability. The **Navigation Day** map, however, displays streets with a wider symbol and with more and larger labels, as illustrated in the following image. You can choose the map that best suits the use case, device, screen size, and so on for your app.

Copyright © 1995-2020 Esri. All rights reserved.

The following example opens a StreetMap Premium mobile map package file and displays the **Navigation Day** map in the app's map view.

```
// Open a StreetMap Premium mobile map package
MobileMapPackage indianaPackage = await MobileMapPackage.OpenAsync(MapPackagePath);

// Get the first map
Map navigationMap = indianaPackage.Maps[0];

// Add it to the map view
MyMapView.Map = navigationMap;
```

⚠️ **Caution:**

While StreetMap Premium data may support some capabilities outside of geocoding, routing, and map display (such as query, spatial analysis, and filtering of individual data layers) it is neither intended nor designed for such uses.

Locate addresses and places

In addition to street data and maps, each StreetMap Premium mobile map package contains a locator task. Use the locator task to geocode addresses, intersections, or places of interest within the area covered by the package.

The following example opens a StreetMap Premium mobile map package file, gets the associated `LocatorTask`, and uses it to find a location.

```
// Open a StreetMap Premium mobile map package
MobileMapPackage indianaPackage = await MobileMapPackage.OpenAsync(MapPackagePath);

// Get the locator task for this area
LocatorTask locatorTask = indianaPackage.LocatorTask;

// Get geocode candidates
IReadOnlyList<GeocodeResult> candidates = await locatorTask.GeocodeAsync("Indianapolis Motor
Speedway");

// Get the candidate with the best score
GeocodeResult match = candidates.OrderByDescending(c => c.Score).FirstOrDefault();
```

For more information on geocoding, see Search for places (geocoding).

Copyright © 1995-2020 Esri. All rights reserved.

Solve routes

Maps in a StreetMap Premium package have an associated transportation network dataset. You can use this dataset to solve routes between two or more locations in the street network. The map must be loaded before you can access the transportation dataset it contains.

The following example gets a `TransportationNetworkDataset` from a map in the StreetMap Premium package, then uses it to create a new `RouteTask`.

```
// Get the (first and only) street network dataset from one of the maps in the package
TransportationNetworkDataset streetNetwork = navigationMap.TransportationNetworks[0];

// Create a new route task that uses the street network
RouteTask routeTask = await RouteTask.CreateAsync(streetNetwork);

// Get the default route parameters from the task
RouteParameters defaultRouteParams = await routeTask.CreateDefaultParametersAsync();

// Create two stops and add them to a list
Stop fromStop = new Stop(fromMapPoint);
Stop toStop = new Stop(toMapPoint);
List<Stop> stops = new List<Stop> { fromStop, toStop };

// Clear any existing stops, then set the new ones
defaultRouteParams.ClearStops();
defaultRouteParams.SetStops(stops);

// Solve the route
RouteResult solveRouteResult = await routeTask.SolveRouteAsync(defaultRouteParams);
```

See Find a route for more information about working with a route task.

Copyright © 1995-2020 Esri. All rights reserved.

Perform analysis

Copyright © 1995-2020 Esri. All rights reserved.

Geoprocessing

Geoprocessing provides a large suite of tools for performing GIS tasks that range from basic buffer and polygon overlays to complex regression analysis and image classification. You can automate all kinds of GIS workflows using Geoprocessing, from converting a large amount of data from one format to another to using a sequence of operations to model and analyze complex spatial relationships.

Examples include:

- Calculating optimum paths through a transportation network
- Predicting the path of wildfire
- Analyzing and finding patterns in crime locations
- Predicting which areas are prone to landslides
- Predicting flooding effects of a storm

There are many reasons why geoprocessing services are good options for these tasks. They can centralize logic to the service. They can help to reduce the workload on the client. Also, geoprocessing services can access tools that aren't available or efficient to run in the client.

Geoprocessing is based on a framework of data transformation. A typical geoprocessing tool performs an operation on an ArcGIS dataset (such as a feature class, raster, or table) and produces a new dataset as a result. Each geoprocessing tool performs a focused operation on geographic data, perhaps as a small part of the overall analytical process.

Geoprocessing allows you to combine a sequence of tools into what is called a model. In a model, the output of one tool is fed into another, which allows you to automate your workflows. Geoprocessing tools and models can be shared with others by creating web services. ArcGIS Runtime uses geoprocessing tools through web services.

A geoprocessing tool is a command or function that performs an operation on GIS data. There are three types of tools, as shown in the table below.

Tool type	Description
Built-in tool	Built-in tools are created by Esri and provided with ArcGIS
Model tool	Model tools are user-created tools authored using ModelBuilder
Script tool	Script tools run a geoprocessing Python script file

Copyright © 1995-2020 Esri. All rights reserved.

Geoprocessing tools can be system tools built by Esri and included in ArcGIS products, or custom tools built as script or model tools by you and other users. You can use both custom and system geoprocessing tools in the same ways. Learn more about using geoprocessing in ArcGIS Desktop by taking a quick tour (in the ArcMap documentation).

Use geoprocessing in ArcGIS Runtime

A geoprocessing web service is published from ArcGIS Desktop to ArcGIS Online or ArcGIS Enterprise. This web service and its tasks can be consumed as REST endpoints by web-based clients such as ArcGIS Desktop, ArcGIS Explorer, custom web apps, and ArcGIS Runtime SDK apps.

Learn more about creating and publishing geoprocessing web services.

What is a geoprocessing task?

A geoprocessing task is a REST child resource of a geoprocessing service. A geoprocessing service can host one or more tasks. A task has a set of parameters that define how a client can interact with the task and what kind of input must be passed to it to execute the task. The server executes the task and returns output values to the client. Output values could include a map service to show the results. This allows for the display of complex data and raster data.

The following figure shows the hierarchy of geoprocessing REST resources. Notice that geoprocessing service properties (shown in orange) are inherited by every geoprocessing task. These include the execution type, the maximum number of records to return, and the map service used to return results.

The format for a geoprocessing task URL is `http://<gpservice-url>/<taskName>`. The endpoint for a task called `BufferPoints` hosted on `myserver`, for example, would be the following.

```
http://myserver/ArcGIS/rest/services/BufferPoints/GPServer/BufferPoints
```

Copyright © 1995-2020 Esri. All rights reserved.

Tasks defined in HTML task pages

The ArcGIS Server service directory contains a set of service pages—web pages of information about each service it provides. You can use the service pages of a geoprocessing service to explore its tasks and their parameters, and learn how to code against them—simply open the REST URL in a browser.

ArcGIS REST Services Directory Login | Get Token

Home > services > Elevation > ESRI_Elevation_World (GPServer) > Viewshed Help | API Reference

JSON

Task: Viewshed

Display Name: Viewshed

Description: Calculates the viewshed of a point given a user defined location and viewing distance.

Category:

Help URL: http://localhost:6080/arcgis/rest/directories/arcgisoutput/Elevation/ESRI_Elevation_World_GPServer/Elevation_ESRI_Elevation_World/Viewshed.htm

Execution Type: esriExecutionTypeSynchronous

Parameters:

Parameter: Input_Observation_Point
Data Type: GPFeatureRecordSetLayer
Display Name Input_Observation_Point
Description: The input location from which the viewshed should be calculated.
Direction: esriGPParameterDirectionInput
Default Value:
Geometry Type: esriGeometryPoint
HasZ: false
HasM: false
Spatial Reference: 54003 (54003)

Fields:
- OBJECTID *(type: esriFieldTypeOID , alias: OBJECTID)*
- OffsetA *(type: esriFieldTypeSmallInteger , alias: OffsetA)*

Features: *None.*

Parameter Type: esriGPParameterTypeRequired
Category:

Parameter: Viewshed_Distance
Data Type: GPLinearUnit
Display Name Viewshed_Distance
Description: The maximum distance from the input point for which the viewshed should be calculated. The maximum allowed distance is 20000 meters.
Direction: esriGPParameterDirectionInput
Default Value: 15000.0 (esriMeters)
Parameter Type: esriGPParameterTypeRequired
Category:

What are task parameters?

Task parameters are the inputs and outputs of a geoprocessing task and vary from task to task based on its individual requirements. Each parameter has a set of properties that describe it, such as name, data type, whether it is required or optional, and so on.

When tasks complete, results are returned in a variety of ways. It's important to understand how a particular geoprocessing task returns results in order to handle them properly. To learn about parameter types, see geoprocessing task parameters.

A geoprocessing task may output a map service result. This prevents the client app from having to symbolize and draw a large number of features or to try to handle a data format that cannot be added directly to the map via the API. When maps from map services are rendered, symbology is applied to the features.

Execution type

The work required of a geoprocessing task can be straightforward and may take a few seconds to execute on the server, or it can support advanced functionality—processing large datasets—and may take longer to execute. Therefore, the REST API provides two execution types, which are options for running the geoprocessing task on the server.

- **Synchronous**: Synchronous tasks are suitable for fast running tasks. In synchronous tasks, the client sends a request to run the task and waits for the results of the task to be returned as a response.
- **Asynchronous**: Asynchronous tasks are suitable for long running jobs. In asynchronous tasks, the server returns a job ID which the client uses to poll for task status. When the status is completed, the results are created as child resources of the job which the client can access using its job ID.

In ArcGIS Runtime both execution types are accessed asynchronously and the call to the correct REST endpoint is defined by the execution type set on the geoprocessing parameters. Even though the geoprocessing tasks are executed in different manner at the service level, they are accessed through one single API. To learn how you can use geoprocessing in your app, see Run a geoprocessing task.

Copyright © 1995-2020 Esri. All rights reserved.

Related topics

Run a geoprocessing task

Copyright © 1995-2020 Esri. All rights reserved.

Run a geoprocessing task

A geoprocessing service contains one or more geoprocessing tasks. A geoprocessing task is a geoprocessing tool running on a service and its execution and outputs are managed by the service. You can run these tools from an ArcGIS Runtime app using classes in the `Esri.ArcGISRuntime.Tasks.Geoprocessing` namespace.

Steps to run a geoprocessing task at a high level:

1. Create a `GeoprocessingTask` using full URL to the target geoprocessing task endpoint.
2. Create `GeoprocessingParameters` providing corresponding `GeoprocessingExecutionType`.
3. Create `GeoprocessingParameter`(s) as required, and add them to `GeoprocessingParameters.Inputs()` where key is the name of the parameter and value is created parameter.
4. Set environment variables on `GeoprocessingParameters` if needed.
5. Create `GeoprocessingJob` by calling the job creation method on the geoprocessing task, using the parameters created previously.
6. Listen for job changed status and message changes on the `GeoprocessingJob`.
7. Run the geoprocessing job and if it completes successfully, get the `GeoprocessingResult`.
8. Handle returned values accordingly. Output parameters are accessed as a dictionary on `GeoprocessingResult` where key is the name of the output parameter and the value is returned parameter. If the geoprocessing service is configured to use a result map service then the result may also contain a map image layer that has been created for you.

Creating a geoprocessing task

`GeoprocessingTask` is the main component when running geoprocessing tasks which is used to run a single geoprocessing task. The target is defined by providing URL to the target REST endpoint.

The format for a geoprocessing task URL is `http://<gpservice-url>/<taskName>`. The endpoint for a task called 'BufferPoints' hosted on 'myserver', for example, would be something like:

https://myserver/ArcGIS/rest/services/BufferPoints/GPServer/BufferPoints

```
GeoprocessingTask hotspotTask =
   new GeoprocessingTask(
      new Uri("http://sampleserver6.arcgisonline.com/arcgis/rest/services/911CallsHotspot/
GPServer/911%20Calls%20Hotspot";));
```

Creating input parameters

`GeoprocessingParameters` contains input parameters and environment variables. Required input parameters are defined by the geoprocessing task. Environment variables are common geoprocessing variables that guide how the service works.

When `GeoprocessingParameters` is created, it needs to know the execution type of the task. This defines which REST endpoint the `GeoprocessingJob` is using when the task is executed. This value needs to match the one defined in the used service.

Copyright © 1995-2020 Esri. All rights reserved.

`GeoprocessingParameters.Inputs()` is a dictionary that contains all the parameters that are sent to the service. The key of the item is the name of the parameter and the value contains the `GeoprocessingParameter`. When parameters are provided to the geoprocessing task their parameter name and type needs to match the one defined in a task. At a minimum all parameters that are set as `required` on the task must be provided to run the task successfully.

You can also provide a set of optional environment variables. These control general settings for the service, such as whether the Z and M values are returned in the result, or which spatial reference to use for the results. For example, if you are visualizing the results of the analysis on the map, requesting the results in the same spatial reference as the map view uses removes the need for the client to reproject the results before adding them to the map view.

```
GeoprocessingParameters hotspotParameters = new
GeoprocessingParameters(GeoprocessingExecutionType.AsynchronousSubmit);


// Construct used query
var queryString = string.Format("(\"DATE\" > date '{0} 00:00:00' AND \"DATE\" < date '{1}
00:00:00')",
    fromDate.ToString("yyyy-MM-dd"),
    toDate.ToString("yyyy-MM-dd"));

// Add query that contains the date range and the days of the week that are used in analysis
hotspotParameters.Inputs.Add("Query", new GeoprocessingString(queryString));
```

Task execution types

When working with geoprocessing services, you must be aware of the execution type that the service was published with. This execution type can be found on the service's REST page. There are multiple execution types that define how geoprocessing tasks run on a server:

- Execute (defined as esriExecutionTypeSynchronous in task page)

  Synchronous execution is designed for fast-running tasks and they are run synchronously on the server. The client sends a request asynchronously to the service and waits for the response. This means these types of services use less resources on the server. When targeting a geoprocessing service that is exposed using this type, `GeoprocessingParameters` needs to be created using `GeoprocessingExecutionType.SynchronousExecute`.

- Submit (defined as esriExecutionTypeAsynchronous in task page)

  Asynchronous services are best suited for long running jobs and are run using asynchronous model on the server. For asynchronous tasks, the server returns a job ID that the client uses to poll for the task status. When the job is completed, the results are fetched separately from the job result endpoints. When targeting a geoprocessing service that is exposed using this type, `GeoprocessingParameters` needs to be created using `GeoprocessingExecutionType.AsynchronousSubmit`.

If a geoprocessing tool is published from ArcGIS Desktop with a map server result to visually represent each geodataset output parameter as a layer then the geoprocessing server (GPServer) will have an accompanying dynamic map server of the same name for presenting the results. This map service's symbology is based on the symbology of the output layer that resulted from running the geoprocessing tool in ArcGIS Desktop during the publishing process. See Defining output symbology for geoprocessing tasks for additional information on setting the symbology for a result map service.

`GeoprocessingJob`, which does the communication between the client and the service, knows which REST endpoint to use based on the execution type that was given as a parameter for `GeoprocessingParameters` and it has to match the execution type on the service. If the value isn't correct then an exception is thrown when the task is run.

Copyright © 1995-2020 Esri. All rights reserved.

Parameters

Geoprocessing tasks usually have input and output parameters. These parameters are defined when the geoprocessing task is authored and can be seen in geoprocessing task page in the services directory. Each parameter of a task has a set of descriptive properties that help you understand the parameter and provide appropriate value for the successful execution of the task.

When parameters are provided to the geoprocessing task their parameter name and type need to match the one defined in a task. At a minimum, all parameters that are set as required on the task must be provided to run the task successfully.

The significance of each property is described below.

Parameter direction

Every parameter has a direction which is defined in the parameter information. This information helps you identify whether the parameter is an input or output parameter.

When the task is executed, the client needs to provide values for the input parameters only. Output parameters are returned from the task when the job has been run successfully.

Output parameters are always returned from a geoprocessing task in a dictionary.

Parameter data types

The data type of the parameter specifies the type of value that the client must provide for the parameter. The value of a parameter can be as simple as a string, long, double, Boolean or date. Or, it can be a set of features or records or a raster file. For example, if the task is expecting to have a raster input parameter, you need to provide a geoprocessing raster parameter in a `GeoprocessingParameters.Inputs()` collection to execute the tool successfully.

Here is a list of ArcGIS Runtime geoprocessing parameter types and which types they correspond to the service.

Required and optional parameters

Parameters can be defined as: required, required with default value, optional, or derived. When the parameter is required but doesn't have default value defined, the parameter must be provided when the geoprocessing task is executed. If the default value is defined, then you can override that value by providing a new parameter in a `GeoprocessingParameters.Inputs()` `dictionary`. Optional parameters can be provided in the same manner. Derived means that the you do not enter a value for the parameter. Derived types are always output parameters.

ArcGIS Runtime parameter types	Corresponding REST types	Note
GeoprocessingString	GPString	A string. If **ChoiceList** is defined in a parameter then the value must match one of the items in the list
GeoprocessingDouble	GPDouble	A decimal value approximately between -2.2E308 to 1.7E308
GeoprocessingLong	GPLong	An integer value between −2,147,483,648 to 2,147,483,647
GeoprocessingBoolean	GPBoolean	True or false

Copyright © 1995-2020 Esri. All rights reserved.

ArcGIS Runtime parameter types	Corresponding REST types	Note
GeoprocessingDate	GPDate	A date value
GeoprocessingLinearUnit	GPLinearUnit	A value that has a distance value and its unit of measurement such as kilometers or meters
GeoprocessingDataFile	GPDataFile	References a file such as `.txt`, `.pdf`, `.csv`, and so on. Supported data types are controlled by the used service. Can reference a file that is uploaded to the service using Uploads capability
GeoprocessingRaster	GPRasterData	References a single raster file such as `.tif` or `.jpg`, and so on. Can reference a raster that is uploaded to the service using Uploads capability.
GeoprocessingRaster	GPRasterDataLayer	Super set of GPRasterData that supports showing values on result map server.
GeoprocessingFeatures	GPRecordSet	A set of **Features** that represents rows and columns. Doesn't have a geometry or spatial reference. Can reference a file that has the features serialized.
GeoprocessingFeatures	GPFeatureRecordSetLayer	A set of **Features** that have a geometry, spatial reference and attributes. Can reference a file that has the features serialized.
GeoprocessingMultiValue	GPMultiValue	An array of geoprocessing parameters of the same data type

Running the task and handling the results

The targeted geoprocessing task is run using `GeoprocessingJob` which is responsible handling the communication between the client and the service. `GeoprocessingJob` is created using `GeoprocessingTask` and `GeoprocessingParameters`, and it can be started by calling `Start`. `GeoprocessingJob` also provides `an awaitable` task which runs the job and returns the results when the task is completed successfully. If any errors occurs while the job is running, an exception is thrown. The error can be also retrieved directly from the `GeoprocessingJob`.

```
// Create job that handles the communication between the application and the geoprocessing task
GeoprocessingJob hotspotJob = _hotspotTask.CreateJob(hotspotParameters);
try
{
    // Execute analysis and wait for the results
    GeoprocessingResult analysisResult = await _hotspotJob.GetResultAsync();

    // Do something with the results
}
catch (TaskCanceledException cancelledException)
{
    // This is thrown if the task is cancelled.
}
catch (Exception exception)
{
    // Handle properly
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Results as a map image layer

The result map server is a dynamic map service that accompanies a geoprocessing service to visualize the geoprocessing results. If this is enabled in the geoprocessing task, then the`GeoprocessingResult.MapImageLayer` might be populated for you. The visualization of the layer is determined by the symbology on the geoprocessing service.

This functionality is available on the geoprocessing services that has execution type of submit with result map server functionality enabled.

```
// Execute analysis and wait for the results
GeoprocessingResult analysisResult = await _hotspotJob.GetResultAsync();
// Add layers to the map view
MyMapView.Map.OperationalLayers.Add(analysisResult.MapImageLayer);
```

In case where `GeoprocessingResult.MapImageLayer` is populated in the results and the geoprocessing result outputs contain `GeoprocessingFeatures` parameters, features aren't automatically downloaded to the client to avoid unnecessary traffic between the service and the client. If you need to download the features, you can get the features explicitly from the results.

```
// Get output features from the results
GeoprocessingFeatures outputFeatures = analysisResult.Outputs["Output_Features"] as
GeoprocessingFeatures;
// Check if the features can be loaded from the service
if (outputFeatures.CanGetOutputFeatures)
{
    // Load features from the service
    await outputFeatures.GetOutputFeaturesAsync();
    foreach (var feature in outputFeatures.Features)
    {
        // For example add features to the graphic overlay
    }
}
```

Downloading geoprocessing data files and rasters

If `GeoprocessingDataFiles` or `GeoprocessingRasters` are used as output parameters, they have a URL specified to the location of the output file. This file can be downloaded to the client device easily by using `an awaitable` task.

```
// Get output raster from the results
GeoprocessingRaster outputRaster = analysisResult.Outputs["Hotspot_Raster"] as
GeoprocessingRaster;
// Download output to the given file path
string fullFilePath = await outputRaster.GetFileAsync("Full file path or folder path");
```

Related topics

Geoprocessing

Copyright © 1995-2020 Esri. All rights reserved.

Author and publish a geoprocessing model

This tutorial will guide you through the process of authoring a geoprocessing model using ModelBuilder in ArcMap, and publishing this model as a geoprocessing package (`.gpk`) to be consumed in an ArcGIS Runtime for .NET WPF application.

Note:
Local Server is required to run a geoprocessing task from a geoprocessing package as described in this topic. In ArcGIS Runtime SDK for .NET, only the Windows desktop (WPF) platform supports Local Server. For more details about running services locally in your ArcGIS Runtime SDK for .NET app, see the Local Server topic.

Download data packages

You will need to download the `SFNetwork.mpk` map package to a directory created for this exercise.

1. Create a new folder named `RoutingExercise` in a location of your choice.
2. Download the SFNetwork.mpk sample data.
3. Copy the `SFNetwork.mpk` map package from the download folder into your `RoutingExercise` folder.
4. Double-click the `SFNetwork.mpk` package in the `RoutingExercise` folder to start ArcMap and unpack the map package's files into your Documents library.

   For example, `...\Documents\ArcGIS\Packages\SFNetwork\v101`.

   Become familiar with the map document and the data it contains.

Create a geoprocessing model

In the following steps, you'll create a geoprocessing model in a new toolbox. You'll add commercial tools to your model, such as the Make Route Layer and Add Locations tools from the Network Analyst Tools toolbox, and configure the model's parameters to use data types supported by ArcGIS Runtime.

1. In the ArcMap **Catalog** window, right-click the `RoutingExercise` folder and click **New** > **Toolbox** to create a toolbox named `RoutingTools`.
2. Right-click the `RoutingTools` toolbox, and click **New** > **Model** to create a model.

   Note:
   The model should be open for editing in ModelBuilder. To add tools to your model, continue with the following steps:

3. From the ArcMap main menu, select **Windows** > **Search** to open the ArcMap **Search** window.
4. Search **Tools** for the Network Analyst Make Route Layer tool.
5. Drag the tool from the search results window to the ModelBuilder window to add an instance of the tool to your model.
6. Double-click the Make Route Layer tool in the ModelBuilder window to open the tool’s dialog box and review its parameters.
7. Click the drop-down arrow to the right of the **Input Analysis Network** field and select your map's Streets_ND network dataset layer.

Copyright © 1995-2020 Esri. All rights reserved.

8. Click OK to accept the default values for the remaining fields and close the **Make Route Layer** dialog box.
9. In the ArcMap Search window, search Tools for the Network Analyst Add Locations tool.
10. Add an instance of the tool to your model by dragging the tool from the search results window to the ModelBuilder window.
11. Use the ModelBuilder Connect tool to connect the Make Route Layer tool's output to the Add Locations tool.
12. Connect the output as the input parameter **Input Network Analysis Layer**.

Modify the model's parameters

For your model to work with WPF apps developed with ArcGIS Runtime SDK for .NET, feature class inputs need to be provided as Feature Set data type parameters. Complete the following steps to designate the Add Locations tool's Input Locations as a model parameter of the required data type:

1. In the ArcMap Catalog window, create a file geodatabase named `FC_Templates` in your `RoutingExercise` folder.
2. Create a point feature class named `RoutePtTemplate` in the geodatabase.
   a. Select Point Features as the geometry type and WGS 1984 as the geographic coordinate system.
   b. Accept the default XY Tolerance, Configuration Keyword, and attribute fields.
   c. Click Finish to create the feature class.
3. In the open ModelBuilder window, right-click the Add Locations tool and select **Make Variable > From Parameter > Input Locations**.
4. Click the **Auto Layout** button then the **Full Extent** button to arrange the model elements in the ModelBuilder window.
5. Right-click the Input Locations variable and click **Model Parameter** on the displayed context menu.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
A bold letter "P" appears next to the variable indicating that it's now a model parameter.

6. Right-click the Input Locations variable and click Properties.
7. Click the **Data Type** tab. The currently specified data type is Table View .
8. Using the Data Type field's pull-down menu, change the data type to **Feature Set** .

Configuring your model to use Feature Set inputs is required to support ArcGIS Runtime development. Feature Sets are in-memory feature classes compatible with the runtime's GPFeatureRecordSetLayer data type.

Define feature set schema and symbology

The attribute schema and spatial reference for a feature set is imported from a template feature class. No particular attribute schema is required for this exercise, so a basic point feature class using the WGS 1984 geographic coordinate system will suffice. Follow these steps to complete the Input Locations Feature Set input's configuration:

1. On the open **Input Locations Properties** dialog box, click the browse button to the right of the **Import schema and symbology from** field.

2. Browse to the `RoutePtTemplate` feature class you created in the `FC_Templates` geodatabase.
3. Click OK to apply the changes to the Input Locations model variable and close the **Properties** dialog box.
4. If you have not already done so, save your model by clicking **Model** > **Save**.

   All of the nodes in your model are now colored indicating that the model is ready to run.

Copyright © 1995-2020 Esri. All rights reserved.

Your feature set's label Input Locations will be used as the input parameter name when an ArcGIS Runtime geoprocessing task is created to consume your model. It is recommended that the names you assign your model variables be descriptive but simple enough for application developers to program against.

Complete the geoprocessing model

The geoprocessing model is nearly complete. Complete the following steps to add the Solve and Select Data tools and configure your model's output parameter:

1. If your model is not already open for editing in ModelBuilder, right-click the model in the ArcMap Catalog window and click **Edit**.
2. In the ArcMap Search window, search Tools for the Network Analyst Solve tool.
3. Add an instance of the tool to your model by dragging the tool from the search results window to the ModelBuilder window.
4. Use the Model Builder Connect tool to connect the Add Locations tool's output to the Solve tool. Connect the output as the input parameter**Input Network Analysis Layer**.

Copyright © 1995-2020 Esri. All rights reserved.

5. Rename the Solve tool's output to `Network Analyst Layer`.
6. Use the ArcMap Search window to search Tools for the ModelBuilder Select Data tool.
7. Add an instance of the tool to your model by dragging the tool from the search results window to the ModelBuilder window.
8. Connect the Network Analyst Layer output as the Select Data tool's **Input Data Element** input parameter.
9. Double-click the Select Data tool in the ModelBuilder window to open the tool's dialog box and review its parameters.
10. Click the drop-down arrow to the right of the **Child Data Element** field to change the selected value from **Stops** to **Routes**. (The Select Data tool allows you to pull a result out of a group result. In this case, the input layer contains the route, barriers, and stops. You'll only use the resulting routes in this exercise.)
11. Click **OK** to close the Select Data tool's parameter window.
12. Right-click the Select Data tool's Routes output and make the output a model parameter.
13. Click the **Auto Layout** > **Full Extent** buttons to arrange the model elements in the ModelBuilder window.
14. From the ModelBuilder main menu, click **Model** > **Model Properties**.
15. Change the **Name** and **Label** properties of your model from **Model** to **Route**.
16. On the Properties dialog box, click the **Parameters** tab and review your model's parameters.

    Your model has only one required input parameter and one derived output parameter. If your model accepted multiple inputs, this is where you see all the parameters, change the order in which scripts and applications are expected to supply their inputs to your model, and verify each parameter's data type.
17. Click **OK** to close the Properties dialog box.
18. Click **Model** > **Save** to save the changes you made to your model.

Copyright © 1995-2020 Esri. All rights reserved.

19. Close the ModelBuilder window.

Execute the model and check the results

To run your model from within ArcMap, complete the following steps. The result can be shared as a geoprocessing package.

1. Zoom to an extent at which you can easily select a point on one of the streets from the **Streets_ND** network dataset.
2. In the ArcMap **Catalog** window, double-click the model to open its interface panel.

The user interface generated by ArcMap indicates that the model is expecting point features as input. Notice that an in-memory feature class, `Route::Input_Locations`, has been temporarily added to your Table of Contents as a new layer. Also, a point feature appears to follow your mouse cursor as you pan across the map. ArcMap is waiting for you to sketch point features as input for the tool.

3. Zoom in if detail from the **Streets**, **Signposts**, or **RestrictedTurns** layers are not displayed; by default, these layers do not display when zoomed-out beyond a certain map scale.
4. Click any two locations on the map display being careful to select points that intersect streets from one of the layers in the network dataset.
5. Once you've sketched two points, click OK on the Route tool's interface to run the geoprocessing model.

Copyright © 1995-2020 Esri. All rights reserved.

If you previously disabled background geoprocessing (**Geoprocessing > Geoprocessing Options**), a modal dialog box will display providing status on the running job. Close this dialog box once the geoprocessing has completed successfully; otherwise, the model will be executed as a background process.

If you previously selected to have geoprocessing results added to the display (**Geoprocessing** > **Geoprocessing Options**), a new feature layer named **Route** will be added to your map display, and you'll see a route that follows the street network connecting the first point you selected (the route start point) to the second selected point (the route end point).

If the route is not displayed on your map, you can add it to the map display by dragging the geoprocessing output to the map display from the ArcMap **Results** window.

- Open the ArcMap Results window (**Geoprocessing** > **Results**).
- Expand the most recent geoprocessing result for the Current Session.
- Select and drag the **Routes: Routes** result to the map display.

Create a geoprocessing package

Once you've verified that your model executes and produces valid results, you can package the model for distribution. Complete the following steps to have ArcMap complete an analysis of your geoprocessing result. This must be done before the result can be published as a geoprocessing package.

1. From the ArcMap main menu, click **Customize > Arcmap Options**.
2. Click the **Sharing** tab and make sure that the **Enable ArcGIS Runtime Tools** option is checked.
3. Click **Geoprocessing > Results** to open the ArcMap Results window.
4. Expand the most recent geoprocessing result for the Current Session.
5. Right-click the result and select **Share As > Geoprocessing Package** from the displayed context menu.

6. On the **Geoprocessing Package** dialog box, select the radio button next to **Save package to file** and browse to your RoutingExercise folder. Name the output package `Route.gpk`.
7. On the Geoprocessing Package dialog, make sure that the **Support ArcGIS Runtime** option is checked.
8. Click the **Analyze** button on the Geoprocessing Package dialog box.

   ArcMap analyzes the geoprocessing result and displays any warnings or errors in a dockable window.

Copyright © 1995-2020 Esri. All rights reserved.

Correct errors and create the geoprocessing package

Complete the following steps to correct any errors. You can ignore warnings from an analysis in ArcMap, but errors must be addressed before you can create the geoprocessing package.

1. In the Prepare window, right-click the **The Route/Input_Locations parameter is missing a syntax dialog explanation** error, and select **Update Tool Metadata** from the displayed context menu.
2. On the **Item Description** dialog box, click the **Edit** button.
3. Scroll down to the **Syntax** field with its input parameter and drop-down arrow.
4. Click the arrow next to the **Input_Locations** parameter to reveal its text entry field.
5. Type the following text in the **Input_Locations** parameter's **Dialog Explanation** field:

   ```
   Model requires at least two input point features which intersect street network features in order to determine a route between the sketched route points.
   ```

6. Save your edit and close the Item Description dialog box.
7. On the ArcMap **Geoprocessing Package** dialog box, click the **Analyze** button to re-analyze the result you are attempting to package. Repeat the previous steps (right-clicking any warnings or errors, working to address the warning or error condition, and then rerunning the analyzer).
   Type the following text in the **Summary (Required)** and **Tags (Required)** fields:

   Summary (Required):

   ```
   Geoprocessing package containing a model which uses commercial Network Analyst tools to complete a point-to-point analysis.
   ```

   Tags (Required):

Copyright © 1995-2020 Esri. All rights reserved.

```
point-to-point, routing, San Francisco
```

8. Once you've addressed all the errors discovered by the analyzer and reviewed the potential impact of ignoring any of the analyzer's warnings, click the **Share** button to create the geoprocessing package.

   Note:
   ArcMap displays a Package dialog box that reports the approximate size of the package being built. This may take several minutes to complete.

You've developed a geoprocessing model within ModelBuilder, run the model within ArcMap, and published the results as a geoprocessing package. Your geoprocessing package, `Route.gpk`, uses Feature Set inputs, so it can be used to support WPF applications developed using ArcGIS Runtime SDK for .NET.

Note:
Local Server is required to run a geoprocessing task from a geoprocessing package as described in this topic. In ArcGIS Runtime SDK for .NET, only the Windows desktop (WPF) platform supports Local Server. For more details about running services locally in your ArcGIS Runtime SDK for .NET app, see the Local Server topic.

Copyright © 1995-2020 Esri. All rights reserved.

Get feature statistics

The statistics query API allows you to get any of the following statistics for a specified field in a feature table: **Sum**, **Average**, **Count**, **Minimum**, **Maximum**, **Standard Deviation**, or **Variance**. In the statistics query parameters, you can define filters for features to include in the statistics (based on attributes, spatial relationships, or time extent) as well as how the results are grouped and sorted.

Follow these steps to query a feature table for statistical information:

1. Load a feature table that supports statistical queries from a local or online data source. This may be a layer shown in a map view in your app, or a table you've loaded but not displayed.

```
// URI for the US states map service
Uri usaStatesServiceUri = new Uri("https://sampleserver6.arcgisonline.com/arcgis/rest/
services/Census/MapServer/3");

// Create the US states feature table
_usStatesTable = new ServiceFeatureTable(usaStatesServiceUri);

// Load the table
await _usStatesTable.LoadAsync();
```

2. Define statistics to return from the query. Create a `StatisticDefinition` for each statistic you want to query from the table. This includes the field name, statistic type, and output alias name for the result.

```
// Create each statistic definition for the query
var statDefAvgPop = new StatisticDefinition("POP", StatisticType.Average, "AveragePop");
var statDefSumPop = new StatisticDefinition("POP", StatisticType.Sum, "TotalPop");
var statDefCount = new StatisticDefinition("OBJECTID", StatisticType.Count, "StateCount");
```

3. Create a `StatisticsQueryParameters` object, passing in your list of (one or more) statistic definitions.

```
// Create a list of the definitions
var statisticDefinitions = new List<StatisticDefinition>
{
    statDefAvgPop, statDefSumPop, statDefCount
};

// Create a new StatisticsQueryParameters object, pass in the list of definitions
var statisticsQueryParams = new StatisticsQueryParameters(statisticDefinitions);
```

4. Define spatial, attribute, or temporal filters (if any) and add them to the query parameters.

```
// Filter the query to only include features in an area of interest
statisticsQueryParams.Geometry = areaOfInterestPolygon;
statisticsQueryParams.SpatialRelationship = SpatialRelationship.Intersects;
```

5. Specify the fields you want to group and order the results with.

```
// Use the SUB_REGION field to group results
statisticsQueryParams.GroupByFieldNames.Add("SUB_REGION");

// Create an OrderBy object to sort on the SUB_REGION field in ascending order
```

Copyright © 1995-2020 Esri. All rights reserved.

```
var orderSubregion = new OrderBy("SUB_REGION", SortOrder.Ascending);
statisticsQueryParams.OrderByFields.Add(orderSubregion);
```

6. Execute the query. The results are returned as a `StatisticsQueryResult`.

```
// Execute the statistical query with these parameters and await the results
StatisticsQueryResult statQueryResult = await
_usStatesTable.QueryStatisticsAsync(statisticsQueryParams);
```

7. Process the query results to read the statistic values. The output alias name defined for each statistic definitions is the key for a corresponding value. If results were grouped, iterate the group names and associated collection of values.

```
var groupedResults = new Dictionary<string, IReadOnlyDictionary<string, object>>();

// Iterate all records in the results and create a dictionary to show in a (hierarchical)
tree view
foreach (StatisticRecord record in statQueryResult)
{
    // Create a name for this group by joining all group field values
    string groupName = string.Join(",", record.Group.Values);

    // Add a dictionary entry with group stats using the group name as the key
    groupedResults.Add(groupName, record.Statistics);
}

// Set the tree view data source
ResultsTreeView.ItemsSource = groupedResults;
```

Tables that support statistical queries

To see if a feature table supports a statistical query, check `ArcGISFeatureLayerInfo.SupportsStatistics` for a feature layer or `ArcGISMapServiceSublayerInfo.SupportsStatistics` for an ArcGIS map image sublayer.

Note:
If the **query** capability is disabled in the service capabilities, then statistics query is also disabled.

When using a service feature table (reading data from an online service, in other words), statistical queries are executed on the service unless the table is in **manual cache** mode, in which case the query executes on the locally available (cached) data. All other feature tables also execute statistical queries against a local data source.

Define a statistics query

Use a `StatisticsQueryParameters` to define:

- Statistics to calculate and their output alias field names
- How results are grouped and ordered
- Filters that restrict which features are evaluated

Each of these are described in the following sections.

Copyright © 1995-2020 Esri. All rights reserved.

Statistic definitions

A `StatisticDefinition` specifies the name of the field that contains values to query, the statistic type, and the output (alias) name to use for the results. A statistics query must contain at least one statistic definition, but may contain several. A field of any data type can be used for statistic types **Count**, **Minimum**, or **Maximum**, which will return the number of values, first value, or last value (when sorted in ascending order) respectively. Other statistic types: **Average**, **Sum**, **Standard Deviation**, and **Variance** are only valid for numeric fields. If these statistics are attempted on a non-numeric field, you will receive an error from the server ("unable to complete operation").

If the output statistic alias name is empty or missing, a default name will be assigned by the map server for the returned statistic field. The server-generated output name is composed of the field name and the statistic type separated by an underscore. If you specify a name, it can only contain alphanumeric characters and an underscore. If the output name is a reserved keyword of the underlying DBMS, the operation may fail.

Group and order results

Unless otherwise specified, the results from a statistical query will not be grouped using a table attribute. In this case, the statistic values returned are single values that describe all the input features. If one or more group fields are specified, the results contain a value for each statistic for each group. If getting average population for world cities grouped by countries, for example, each country would have a value for the average population of their cities. Without this grouping, you would get a single value for all cities in the world.

Results are also not ordered unless you specify one or several fields to sort on. The fields you choose to order results by must be fields you've also chosen for grouping. The following example shows results for statistics on US states grouped by sub-region. The results are also sorted by sub-region (in ascending order).

```
▷ East North Central
▷ East South Central
▷ Middle Atlantic
◢ Mountain
      AGE_30_39_Maximum : 761246
      POP2000_Sum : 18172295
      POP2007_Sum : 21492235
      STATE_NAME_Count : 8
▷ New England
◢ Pacific
      AGE_30_39_Maximum : 5500264
      POP2000_Sum : 45025637
      POP2007_Sum : 49731702
      STATE_NAME_Count : 5
▷ South Atlantic
▷ West North Central
▷ West South Central
```

Filter features to evaluate

You can define any combination of spatial, attribute, or temporal criteria to filter the features that are evaluated by the statistical query. If none of these filters are set, the query will be evaluated against all features in the table.

- **Attribute**—define a where expression that uses any of the fields in the table.
- **Spatial**—use a geometry object and a spatial relationship.
- **Temporal**—specify a time or time range.

Copyright © 1995-2020 Esri. All rights reserved.

Process results from a statistical query

The results from a statistical query are returned as an enumerator of `StatisticRecord` objects. Each record describes a group and contains a dictionary of group values and a dictionary of statistic values. The group dictionary contains key/value pairs that describe each group field, with the field name ("SUB_REGION", for example) as the key for the field value ("Mountain", for example). Since multiple group fields can be specified for a query, each record in the results represents a unique combination of the group field values.

The statistics dictionary for each record contains the requested statistics for the group. The key for each statistic value is the alias name (by default, in the form **fieldName_statisticType**). If no group fields were specified for the query, the groups dictionary is empty and a single set of statistics are returned to describe all features included in the query.

Copyright © 1995-2020 Esri. All rights reserved.

Make measurements

Just as there are many types of projections available to display the earth's surface on a two-dimensional map, there are a variety of methods for measuring distance or area. As when choosing a projection, you need to understand the characteristics of measurement methods when making measurements for a particular use.

The following are some questions to ask when working with geographic measurements:

- Will measurements be at a large or small scale? The curvature of the earth's surface is not as much of a factor for measurements made at a larger scale (smaller areas).
- What coordinate system will be used? With a projected coordinate system, the properties of the underlying projection greatly influence the accuracy of measurements in a given area of the map.
- How accurate do measurements need to be? Sometimes you want to interactively explore using approximate measurements. At other times, a high level of accuracy is required.
- Will measurements include things above or below the surface of the earth? Some measurement methods do not consider the z-value (elevation or altitude) for the geometry being measured.
- Should measurements take topography into account? Distance for a hiking trail, for example, is more accurate if it accounts for travel across the surface terrain.
- Are measurements being made for travel on a predefined line network (representing roads or trails, for example)? Routing allows you to model this kind of travel and make measurements in distance as well as time or other costs.

Planar versus geodetic measurements

Planar measurements are made with straight lines in two dimensions. If you're using a geographic coordinate system, planar measurements must cut through the curved surface of the earth rather than following it (like pressing a ruler into a globe). This causes the measurement to be shorter than the true distance, and the error becomes greater as the length of the line increases.

Planar measurements using a projected coordinate system are like placing a ruler on a paper map. In this case, the properties of the underlying projection influence the accuracy of measurements in a given area. A Web Mercator projection, for example, exaggerates areas of the map that are not near the equator. Measurements will, therefore, be most accurate near the equator and error will increase as you move toward the poles. Planar measurements are generally best if made over small areas in a suitable spatial reference, such as the correct UTM zone for the area.

⚠ Caution:

Planar measurements of distance and area can be extremely inaccurate if using an unsuitable spatial reference. Make sure you understand the potential for error and the effect of spatial references if you need accurate results.

Measurements relative to a curved surface (nonplanar) are generically referred to as geodetic. Although not an exact representation of the earth, and measurements do not follow actual features on the surface (terrain, in other words), geodetic measurements are more accurate than planar measurements, especially over long distances.

Copyright © 1995-2020 Esri. All rights reserved.

For example, the following image shows distance measurements between Reykjavik and Saint Petersburg in a Web Mercator map. The red line is a planar measurement and the light gray line is geodetic. If a similar measurement were made near the equator, the values would be much closer.

The following are a variety of geodetic measurement types:

- Great circle—Any circle or near circle produced by the intersection of the surface of a sphere and a flat plane that passes through the center of the sphere. The equator and all lines of longitude are great circles. Great circles are used in navigation, since the shortest path between two points on the earth's surface lies on a great circle.
- Geodesic—The shortest distance between two points on the surface of a spheroid (also known as an ellipsoid). For example, the shortest path between two points along a meridian form a geodesic. This is similar to the great circle method, which models the surface as a sphere.

> **Note:**
> It's easy to confuse the terms geodetic and geodesic. Remember that geodetic refers to any measurement that uses a curved model of the earth's surface. Geodesic measurements specifically use an ellipsoid to model the shape of the earth.

- Normal section—A normal section is a simplified version of a geodesic line. It's defined by the intersection of a plane that passes through two points on the surface of the spheroid and is perpendicular to the surface (normal) at the first point.
- Loxodrome—A line between two locations at a constant bearing is known as a loxodrome (or rhumb line). Loxodromes do not attempt to calculate the shortest distance since the line between the points is defined by the bearing. However, they have historical significance for nautical navigation, where a ship or plane could follow a constant compass bearing from one point to another. Only when a loxodrome is oriented precisely north-south or east-west does it define a geodesic (shortest path) line.

The geodetic methods described previously account for the earth's curvature, which makes them well-suited for nautical and aeronautical routes. However, they ignore surface topography and thus are not ideal for measuring land travel over terrain with a lot of relief. Surficial measurement, on the other hand, can be used to measure distance as it would be traveled along the terrain. With ArcGIS Pro and ArcGIS Desktop, for example, you can use the Add surface information tool to add length or area fields to a vector dataset whose values are calculated from an input elevation surface. Geoprocessing also provides a way to implement custom measurement tools.

Similarly, calculating routes along a transportation network is the most accurate way to measure travel (distance and time) over roads or trails. For information about modeling travel across a network, see the Find a route topic.

Copyright © 1995-2020 Esri. All rights reserved.

Use the geometry engine to make measurements

`GeometryEngine` provides several methods for making planar or geodetic measurements of distance or area. These measurements are made independent of the display (no map or scene is necessary). Planar measurements use the spatial reference and units of the input geometry. Even if in a geographic coordinate system, planar measurements do not consider the curvature of the earth and will return values in decimal degrees. You should, therefore, project measure geometry to an appropriate projected coordinate system before making planar measurements.

⚠ Caution:

Measurements made with `GeometryEngine` do not account for the z-value (elevation or altitude) of the input geometry. Whether planar or geodetic, measurement locations are always assumed to be on the surface of the earth.

The geometry engine methods `Area`, `Distance`, and `Length` calculate planar measurements. They have the geodetic equivalents `AreaGeodetic`, `DistanceGeodetic`, and `LengthGeodetic`. If a geometry engine method doesn't contain the word geodetic, you can assume it's a planar operation.

The following code makes a planar distance measurement between Edinburgh and Dar es Salaam. The input locations are first projected to a world equidistant projection.

```
// Define geographic map points for the cities of Edinburgh (Scotland) and Dar es Salaam
(Tanzania).
_edinburghGeographic = new MapPoint(-3.1883, 55.9533, 0.0, SpatialReferences.Wgs84);
_darEsSalaamGeographic = new MapPoint(39.2083, -6.7924, 0.0, SpatialReferences.Wgs84);

// Create a world equidistant cylindrical spatial reference for measuring planar distance.
SpatialReference equidistantSpatialRef = SpatialReference.Create(54002);

// Project the points from geographic to the projected coordinate system.
MapPoint edinburghProjected = GeometryEngine.Project(_edinburghGeographic,
equidistantSpatialRef) as MapPoint;
MapPoint darEsSalaamProjected = GeometryEngine.Project(_darEsSalaamGeographic,
equidistantSpatialRef) as MapPoint;

// Get the planar distance between the points in the spatial reference unit (meters).
double planarDistanceMeters = GeometryEngine.Distance(edinburghProjected, darEsSalaamProjected);
// Result = 7372671.29511302 (7,372.6713 kilometers)
```

When making geodetic measurements with the geometry engine, the type of geodetic measurement can be specified: geodesic, great elliptic, normal section, or loxodrome.

The following code illustrates making several geodetic distance measurements between Edinburgh and Dar es Salaam. The result of each measurement is shown in the comments at the end. With the exception of loxodrome, the values are within a meter. Compare these to the planar result, which is nearly 600 kilometers shorter.

```
// Get the geodesic measurement between the points in kilometers.
GeodeticDistanceResult geodesicMeasureResult = GeometryEngine.DistanceGeodetic(
            _edinburghGeographic,
            _darEsSalaamGeographic,
            LinearUnits.Kilometers,
            AngularUnits.Degrees,
            GeodeticCurveType.Geodesic);

// Get the great elliptic measurement between the points in kilometers.
GeodeticDistanceResult greatEllipticMeasureResult = GeometryEngine.DistanceGeodetic(
    _edinburghGeographic,
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    _darEsSalaamGeographic,
    LinearUnits.Kilometers,
    AngularUnits.Degrees,
    GeodeticCurveType.GreatElliptic);

// Get the normal section measurement between the points in kilometers.
GeodeticDistanceResult normalSectionResult = GeometryEngine.DistanceGeodetic(
    _edinburghGeographic,
    _darEsSalaamGeographic,
    LinearUnits.Kilometers,
    AngularUnits.Degrees,
    GeodeticCurveType.NormalSection);

// Get the loxodrome measurement between the points in kilometers.
GeodeticDistanceResult loxodromeMeasureResult = GeometryEngine.DistanceGeodetic(
    _edinburghGeographic,
    _darEsSalaamGeographic,
    LinearUnits.Kilometers,
    AngularUnits.Degrees,
    GeodeticCurveType.Loxodrome);

// Measurement results:
// - geodesicMeasureResult.Distance = 7,965.9256 Km
// - greatEllipticMeasureResult.Distance = 7,965.9265 Km
// - normalSectionResult.Distance = 7,965.9264 Km
// - loxodromeMeasureResult.Distance = 8,008.0699 Km
```

A geodetic distance measurement result provides the following information:

- Azimuth1—Azimuth from the first location toward the second. This value is in the chosen angular unit.
- Azimuth2—Azimuth from the second location toward the first. This value is in the chosen angular unit.
- Azimuth unit—Angular unit used to measure the azimuth values between input locations.
- Distance—Geodetic distance value between the input locations. This value is in the chosen linear unit.
- Distance unit—Linear unit used to measure the distance between input locations, such as meters, feet, kilometers, miles, and so on.

The geometry engine can also calculate geodetic area measurements and buffers. The following image shows geodetic (red) and planar (blue) polygons produced by buffering at a standard distance. Although, in this Web Mercator projection, the geodesic polygons appear larger as you move away from the equator; the area they contain is the same. Some projections can distort area, especially as you move away from the parallel or meridian on which they're based (the equator in this example), and, therefore, may not be suited for planar measurements.

Copyright © 1995-2020 Esri. All rights reserved.

Use location distance measurement analysis in a scene view

At version 100.2, ArcGIS Runtime introduced analysis and analysis overlay classes to provide fast and dynamic analysis results in a scene view. The `Analysis` classes allow you to define analyses (such as viewshed, line of sight, and measure) that use data in the current scene and render results dynamically in the view. Analysis results are displayed using an analysis overlay in a scene view. For more information about working with analysis overlays, see Manage analysis with analysis overlays.

You can use the `LocationDistanceMeasurement` analysis class to make interactive distance measurements between two locations in the scene view. Because the input locations are 3D points, you can measure between locations on, above, or below the surface.

Note:
Because the measurements are planar, accuracy is good only for relatively short distances. Terrain is not considered in the measurement.

A location distance measurement returns the following distances:

- Direct distance—The straight-line distance between the start and end locations
- Horizontal distance—The planar distance between the start and end locations as projected onto the surface of the earth
- Vertical distance—The difference in elevation (z-value) between the start and end locations

As you update the measurement end location (perhaps by dragging on the display), the analysis displays a graphic with lines for direct, horizontal, and vertical measurements.

Copyright © 1995-2020 Esri. All rights reserved.

Caution:

A location distance measurement analysis only calculates distances when it's contained in an analysis overlay that is in the scene view's analysis overlay collection. When you initially create a location distance measurement analysis, distances remain undefined until added to an analysis overlay in the scene view. If removed from the analysis overlay collection, the last distances are retained and are not updated.

Use the following steps to add a location distance measurement analysis to a scene view:

1. Create a new `LocationDistanceMeasurement`, and set the start and end locations. These locations must be defined using 3D points (those with x-, y-, and z-coordinate values). If allowing the user to define these points interactively, you can define the locations from tap events on the scene view.

```
// Set up the scene analysis measure tool.
LocationDistanceMeasurement distanceMeasurement = new LocationDistanceMeasurement();
```

```
// Provide the start and end location points.
distanceMeasurement.StartLocation = cityCenterPoint;
distanceMeasurement.EndLocation = roofTopPoint;
```

2. Set the unit system for distance results: metric (meters and kilometers) or imperial (feet and miles).

```
// Set the unit system for the measurements (metric or imperial).
distanceMeasurement.UnitSystem = Esri.ArcGISRuntime.UnitSystem.Metric;
```

Copyright © 1995-2020 Esri. All rights reserved.

3. Handle the measurement-changed event. When the measurement changes, you can read updated distances. Measurement values will be in the appropriate unit (for the chosen unit system), depending on the length: meters or feet for shorter measurements (under 3,000 meters or 1,000 feet) and kilometers or miles for those that are longer.

```
// Handle the MeasurementChanged event for the measurement analysis.
distanceMeasurement.MeasurementChanged += (s, evt) =>
{
    // Strings to report each distance.
    string directDistanceText = "undefined";
    string horizontalDistanceText = "undefined";
    string verticalDistanceText = "undefined";

    // Create a formatted string that shows the direct distance value and units.
    if (distanceMeasurement.DirectDistance != null)
    {
        double directDistanceValue = distanceMeasurement.DirectDistance.Value;
        string directDistanceUnit = distanceMeasurement.DirectDistance.Unit.Abbreviation;
        directDistanceText = string.Format("{0:N} {1}", directDistanceValue,
directDistanceUnit);
    }

    // Create a formatted string that shows the horizontal distance value and units.
    if (distanceMeasurement.HorizontalDistance != null)
    {
        double horizontalDistanceValue = distanceMeasurement.HorizontalDistance.Value;
        string horizontalDistanceUnit =
distanceMeasurement.HorizontalDistance.Unit.Abbreviation;
        horizontalDistanceText = string.Format("{0:N} {1}", horizontalDistanceValue,
horizontalDistanceUnit);
    }

    // Create a formatted string that shows the vertical distance value and units.
    if (distanceMeasurement.VerticalDistance != null)
    {
        double verticalDistanceValue = distanceMeasurement.VerticalDistance.Value;
        string verticalDistanceUnit = distanceMeasurement.VerticalDistance.Unit.Abbreviation;
        verticalDistanceText = string.Format("{0:N} {1}", verticalDistanceValue,
verticalDistanceUnit);
    }

    // Show the distances in the UI.
    DirectTextBlock.Text = directDistanceText;
    HorizontalTextBlock.Text = horizontalDistanceText;
    VerticalTextBlock.Text = verticalDistanceText;
};
```

4. Create an analysis overlay with the location distance measurement and add it to the scene view. The distances are not initialized or updated unless the location distance measurement analysis is in an analysis overlay in the scene view's analysis overlay collection.

```
// Create an analysis overlay.
AnalysisOverlay sceneAnalysisOverlay = new AnalysisOverlay();

// Add the LocationDistanceMeasurement to the analysis overlay.
sceneAnalysisOverlay.Analyses.Add(distanceMeasurement);
MySceneView.AnalysisOverlays.Add(sceneAnalysisOverlay);
```

You can add several distance measurement analyses to a scene view and manage their visibility by turning them on or off.

Copyright © 1995-2020 Esri. All rights reserved.

Related topics

Analyze visibility in a scene view
Manage analyses with analysis overlays

Copyright © 1995-2020 Esri. All rights reserved.

Manage analyses with analysis overlays

At version 100.2, ArcGIS Runtime introduced **analysis** and **analysis overlay** classes to provide fast and dynamic visualization of analysis results in a scene view. The `Analysis` classes allow you to define analyses (such as viewshed, line of sight, or distance measurement) to be performed using data in the current scene, then render results that are updated dynamically (as layers are added or removed, features are updated, camera position changes, and so on). See the Analyze visibility in a scene view and Location distance measurement topics for examples of analysis classes.

Add an analysis overlay

One or more analysis objects can be added to an analysis overlay, which is then added to the scene view. The analysis overlay can control result visibility for all analyses it contains, allowing you to manage a set of related analyses as a single overlay displayed in the view. A scene view can contain several analysis overlays.

⚠️ **Caution:**

Some analysis types enforce a maximum number that can be added to a single scene view (the value can vary on mobile devices and desktop platforms). See the documentation for specific analyses to see if a maximum number of analyses exists.

Use the following steps to add an analysis overlay to your scene view:

1. Create a new `AnalysisOverlay` to contain one or more analyses. You might use one overlay to contain all the line of sight analyses for a specific observation point, for example.

```
AnalysisOverlay allLinesOfSightOverlay = new AnalysisOverlay();
```

2. Add the analysis overlay to the scene view's overlay collection. Similar to the graphics overlay collection, a scene view can have zero or several analysis overlays in the collection.

```
MySceneView.AnalysisOverlays.Add(allLinesOfSightOverlay);
```

3. Add one or more analysis objects to the overlay. You can add and remove analyses in the overlay whenever you need to.

Copyright © 1995-2020 Esri. All rights reserved.

```
allLinesOfSightOverlay.Analyses.Add(rooftopToIntersection);
allLinesOfSightOverlay.Analyses.Add(rooftopToWindow);
```

Controlling analysis visibility

Visibility for scene analyses can be toggled on or off for individual analysis classes (a single line of sight analysis, for example) or for an entire analysis overlay. Toggling the visibility for an analysis overlay will affect all the analyses it contains. This allows you to organize your analyses into logical groups and manage their visibility as a single unit.

You can toggle visibility for an individual analysis to show/hide its results, or toggle visibility for the analysis overlay (and all analyses it contains) as a group. You can also remove it from the analysis overlay if you no longer need it, as well as remove the entire overlay from the scene view.

```
// Hide an analysis display
rooftopToWindow.IsVisible = false;

// Show the analysis overlay display (to render all analyses that are also set to visible)
allLinesOfSightOverlay.IsVisible = true;

// Remove an analysis from the overlay
allLinesOfSightOverlay.Analyses.Remove(rooftopToIntersection);

// Remove the analysis overlay from the scene view analysis overlay collection
MySceneView.AnalysisOverlays.Remove(allLinesOfSightOverlay);
```

Related topics

Analyze visibility in a scene view
Make measurements

Copyright © 1995-2020 Esri. All rights reserved.

Analyze visibility in a scene view

Visibility analysis means determining which areas or specific features are visible from a hypothetical observer (location, heading, field of view, and so on). This can help answer questions like: Which apartments have a view of the beach? Which areas of the forest are visible from a car driving through it? Does anything block the view between my office window and my parking spot?

For evaluating visibility in a scene view, ArcGIS Runtime provides `Viewshed` and `LineOfSight` classes. Viewshed analysis highlights **areas** in the scene that are visible from a given observer. Line of sight shows which segments are visible along a **line** drawn between an observer and a target location. In either type of analysis, the observer may be moving or stationary.

For visibility analysis, elevation surfaces, scene layers (buildings, for example), and 3D graphics (such as models) are considered obstructions to visibility. Billboarded (flat) graphics, such as text and simple graphics are not considered in the analysis. Visibility analysis is reactive, meaning that as data are added to (or removed from) the scene, visibility will be reevaluated and updated dynamically in the display.

Scene analyses versus geoprocessing

At version 100.2, ArcGIS Runtime introduced analysis and analysis overlay classes to provide fast and dynamic (but strictly visual) analysis results in a scene view. The `Analysis` classes (including the visibility analysis classes described in this topic) allow you to define analyses to be performed using data in the current scene, then render results that are updated dynamically (as layers are added or removed, features are updated, camera position changes, and so on). Analysis results are displayed using an analysis overlay. For more information about working with analysis overlays, see Manage analysis with analysis overlays.

Scene analyses produce results that are strictly visual, rendered dynamically and quickly by the GPU. As analyses are evaluated, results are rendered in the scene view for display only. This is ideal for use cases that require interactive and visual evaluation of results that change frequently. However, there are no features (such as graphics) or output geometry produced, so use cases like persisting result shapes or performing hit tests with other features are not supported.

Scene analysis uses the data currently available in the scene view to perform analysis (evaluate visibility, for example) and then quickly render the result. Since the data in the scene view changes according to the level of detail (distance of the camera to the surface, in other words), the analysis results may change as you interact with the scene view (zoom in or out, change camera tilt, and so on). Resolution of surface elevation data varies according to level of detail and for visibility analysis will affect the result, as shown in the following image.

Copyright © 1995-2020 Esri. All rights reserved.

Geoprocessing provides another way to perform analysis using a suite of specialized GIS analysis tools that run on a service. The results returned from geoprocessing vary by task, but may include a map image, output features, raster images, data files, and so on. Geoprocessing usually gives you the ability to access the resulting features (or geometry) from the task, allowing you to save your results or use them for further analysis (find all the points inside the visible area, for example). Geoprocessing generally requires several inputs from the client, and takes some time to execute and return results from the server. See Geoprocessing to learn more.

As a general guide, scene analysis allows you to interactively explore data visually, but lacks accuracy and does not provide tangible results. Geoprocessing provides better accuracy, gives you more control over the analysis, and can provide result features or geometry, but is not well suited for quickly and interactively exploring a variety of scenarios.

There are samples that illustrate viewshed analysis using both techniques: Analyze viewshed (geoprocessing), Viewshed from camera (scene analysis).

Viewshed analysis

Viewsheds are a type of visibility analysis that show the visible and obstructed areas within a directed field of view. Viewshed analysis shows you the areas of the scene that are visible from a given observer, determined by the terrain (represented by an elevation surface), buildings and other 3D features (represented by scene layers, graphics, and so on), and the properties of the observer.

The 3D shape inside which visibility is evaluated is called the **frustum**. The frustum is defined by the observer's location, horizontal and vertical angles of vision, heading, pitch, and so on. You can think of the frustum as the full volume being evaluated within the observer's field of vision.

Copyright © 1995-2020 Esri. All rights reserved.

The following properties define the frustum:

- **Heading**—a value in degrees (azimuth) that defines the direction of the observer's line of vision. This is the direction of the center of the frustum. The value is normalized to between 0 - 360.
- **Minimum distance**—the distance (meters) from the observer at which viewshed analysis begins.
- **Maximum distance**—the distance (meters) from the observer beyond which the viewshed is not evaluated.
- **Horizontal angle**—an angle that defines the observer's horizontal field of vision. The angle is split evenly on the right and left sides of the heading. This value must be greater than zero to produce a valid frustum, and no greater than the maximum of 360 degrees.
- **Vertical angle**—an angle that defines the observer's vertical field of vision. The angle is split evenly on the top and bottom sides of the heading. This value must be greater than zero to produce a valid frustum, and no greater than the maximum of 360 degrees.
- **Pitch**—the angle of the observer's tilt relative to the surface. Pitch does not affect the shape of the frustum, but tilts it relative to the surface. Pitch values must be between 0 (looking straight down towards the center of the earth) and 180 (looking straight up).

Note:

The area between the observer and the near plane of the frustum (defined by the minimum distance) is **not** included in the viewshed analysis. If an object exists in that area (a building or hill, for example), it will not be considered an obstruction to visibility. In other words, only the space represented by the frustum is analyzed for visibility. In general, the minimum distance value should be used to prevent nearby objects from interfering with the analysis (such as the corner of a building, for example) and should be relatively small. However, you could use a larger minimum distance to answer questions like "what would the view from my window be like if the neighbor's house wasn't there?".

The frustum can be displayed as part of the viewshed analysis result, which may be helpful for troubleshooting or verifying the observer properties. To toggle the visibility of the frustum on or off, use the `IsFrustumOutlineVisible` property. The default frustum outline symbol is a fully opaque blue line, but you can specify a custom color if you like. The frustum color you provide will be applied to all viewshed analyses in the scene view.

Viewshed results are displayed as two visual classes in the scene view: one representing visible areas, and one representing areas of obstructed visibility. By default, these are rendered green and red respectively, but you can specify your own colors. Areas that are not symbolized for visibility are outside of the frustum and therefore not evaluated. In the image below, the green areas are visible and the red areas are obstructed. Everything else is outside of the frustum.

As with the frustum color, the colors you provide for visible and obstructed areas will be applied to all viewshed analyses in your scene view. These colors cannot be set individually on each analysis, in other words. If adding several viewshed analyses to your scene view, transparency in your visibility colors will help visualize overlapping areas by making them appear darker.

Caution:

A scene view can support a limited number of viewshed analyses. The actual number will vary according to the available memory on your device and the size of viewshed frustums but will typically be between three and seven for mobile devices and up to fifteen or so on a desktop machine. If you exceed the number of viewsheds your device can display, they will cease rendering in the scene view.

Copyright © 1995-2020 Esri. All rights reserved.

The observer for viewshed analysis may be represented by a point (x, y, and z coordinate values) or by a `GeoElement` with point geometry.

Location viewshed

Use the `LocationViewshed` to define the observer using a point location. You also need to provide heading, pitch, and roll, the vertical and horizontal angles of vision, and minimum/maximum distance of the viewshed analysis to define the frustum. You can dynamically update the observer for the viewshed using a camera object that encapsulates these properties of an observer.

To add `LocationViewshed` analysis to your scene view:

1. Create a new `LocationViewshed`. In the constructor, pass values that define the observer (frustum): location, heading, pitch, horizontal and vertical angles, and the minimum and maximum distance. Optionally, you could create a camera to represent the observer, and pass that to the constructor along with minimum and maximum distance.

```
// create a point for the observer location
MapPoint observerPoint = new MapPoint(-117.1979, 34.1409, 500.0, SpatialReferences.Wgs84);

// define the heading, pitch, and roll
double headingAzmuth = 270.0;
double pitchAngle = 75.0;
double rollAngle = 0.0;

// define horizontal/vertical field of vision and minimum/maximum distance for visibility
(meters)
double horizontalViewAngle = 82.5;
double verticalViewAngle = 80.0;
double minViewDistanceMeters = 0.0;
double maxViewDistanceMeters = 2000.0;

// create a location viewshed analysis with values that define the frustum
var camera = new Camera(observerPoint, headingAzmuth, pitchAngle, rollAngle);
LocationViewshed viewFromLocation = new LocationViewshed(camera, minViewDistanceMeters,
maxViewDistanceMeters);
// .. or ..
// LocationViewshed viewFromLocation = new LocationViewshed(observerPoint, headingAzmuth,
pitchAngle, horizontalViewAngle, verticalViewAngle, minViewDistanceMeters,
maxViewDistanceMeters);
```

2. (**Optional**) Assign custom colors for analysis results: visible areas, obstructed areas, and frustum. To show the frustum in the results, set `IsFrustumOutlineVisible` to **true**.

```
// define new colors for the frustum outline and for visible/obstructed areas in the viewshed
result
Color visibleColor = Color.FromArgb(255, 222, 0, 255);
Color obstructedColor = Color.FromArgb(150, 255, 99, 71);
Color frustumColor = Color.FromArgb(255, 70, 130, 180);
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// these are static properties on the Viewshed base class that apply to all viewshed analyses
in the scene view
Viewshed.VisibleColor = visibleColor;
Viewshed.ObstructedColor = obstructedColor;
Viewshed.FrustumOutlineColor = frustumColor;

// show the frustum outline for this analysis
viewFromLocation.IsFrustumOutlineVisible = true;
```

3. Add the analysis to an `AnalysisOverlay` to display the results (as described in Manage analyses with analysis overlays).
4. You may update the analysis after it's associated with your scene view by changing the observer position or result colors. A convenience method allows you to update the observer using a camera.

```
viewFromLocation.UpdateFromCamera(newObserverCamera);
```

GeoElement viewshed

You may want to see a viewshed for an observer whose location, heading, or pitch update frequently. For example, you may want to display a viewshed for the pilot of a search plane as it flies over the terrain. As an element representing the aircraft changes its position in the scene view, you'd like the viewshed result to update accordingly. This can be accomplished using `GeoElementViewshed` scene analysis, which takes a `GeoElement` (of point geometry type) to define the observer's position. Since the viewshed is based on the `GeoElement`, any change in its position (location, heading, pitch, and so on) will update the viewshed result. You can also specify offset values for the observer relative to the element. This is useful for more accurately representing an observer position inside a moving craft, for example.

Note:
A `GeoElement` can be a `Feature` or a `Graphic`. See Features and graphics for more information.

To add a `GeoElementViewshed` analysis to your scene view:

1. Create a point `GeoElement` to represent the observer in the viewshed. Add the element to the scene view.
2. Create a new `GeoElementViewshed`. In the constructor, define the observer by passing in the `GeoElement`, horizontal and vertical field of view, minimum and maximum distance, and heading and pitch offsets.

```
// define horizontal/vertical field of vision and minimum/maximum distance for visibility
(meters)
double horizontalViewAngle = 40.0;
```

Copyright © 1995-2020 Esri. All rights reserved.

```
double verticalViewAngle = 60.0;
double minViewDistanceMeters = 0.0;
double maxViewDistanceMeters = 5000.0;

// define heading and pitch offsets (if any)
double headingOffset = 0.0;
double pitchOffset = 0.0;

// create the viewshed analysis using a GeoElement (Graphic) to define the observer
GeoElementViewshed viewFromPlane = new GeoElementViewshed
    (_planeGraphic,
    horizontalViewAngle,
    verticalViewAngle,
    minViewDistanceMeters,
    maxViewDistanceMeters,
    headingOffset,
    pitchOffset);
```

3. [**Optional**] Define offset values to define the X, Y, and Z location of the observer relative to the element. These offsets are defined in meters using a coordinate system relative to the center of the object, where X is to the right or left, Y is forward or back, and Z is up and down. To place an observer in the driver's seat of a car, for example, you might place her 60 centimeters from the left of the car center, 70 centimeters forward from center, and 40 centimeters above the center. These offset values would be (-0.6, 0.7, 0.4).

```
// apply x/y/z offsets (meters) to accurately place the observer
viewFromPlane.OffsetX = 2.5;
viewFromPlane.OffsetY = 0.0;
viewFromPlane.OffsetZ = 1.0;
```

4. (**Optional**) Provide custom colors for analysis results: visible areas, obstructed areas, and frustum. To show the frustum in the results, set `IsFrustumOutlineVisible` to **true**.

```
// define new colors for the frustum outline and for visible/obstructed areas in the viewshed
result
Color visibleColor = Color.FromArgb(255, 222, 0, 255);
Color obstructedColor = Color.FromArgb(150, 255, 99, 71);
Color frustumColor = Color.FromArgb(255, 70, 130, 180);

// these are static properties on the Viewshed base class that apply to all viewshed analyses
in the scene view
Viewshed.VisibleColor = visibleColor;
Viewshed.ObstructedColor = obstructedColor;
Viewshed.FrustumOutlineColor = frustumColor;

// show the frustum outline for this analysis
viewFromPlane.IsFrustumOutlineVisible = true;
```

5. Add the analysis to an `AnalysisOverlay` to display the results (as described in the Manage analyses with analysis overlays topic).

As the position of the `GeoElement` changes, the viewshed results will update accordingly.

Copyright © 1995-2020 Esri. All rights reserved.

Line of sight analysis

Line of sight analysis shows visibility along a line drawn between an observer and a target location. The result shows which segments of the line can be seen by the observer and which segments are blocked by an obstruction. In the image below, the cyan segment of the line is the portion of the sight line that is not blocked, while the magenta segments are obstructed by the terrain (and therefore not visible).

In addition to **visible** and **obstructed** line segments, you may see portions of the line that have an **unknown** visibility state. These are lines drawn in gray to indicate that visibility cannot be reliably evaluated with the currently available data. For example, visibility may become unknown if the camera position is changed such that the target and/or observer for the line of sight are no longer displayed. In that situation, data outside the visible display may have been unloaded from the scene. The unloaded data may have contained obstructions to the line of sight that are no longer present. Obstructed lines, however, may still render as obstructed when the target or observer moves off the display, since the obstruction is assumed to still exist even if the data have unloaded.

A line of sight analysis represents a single line from observer to target. To make sight lines from multiple vantage points, you must create multiple analysis instances. A group of related line of site analyses (for the same observer, for example) can be managed in a single analysis overlay (see Manage analyses with analysis overlays).

Copyright © 1995-2020 Esri. All rights reserved.

Caution:

A scene view is restricted to a maximum of sixty-four line of sight analyses. Be aware of performance considerations when adding many analyses to a scene view.

Location line of sight

Use the `LocationLineOfSight` to define the observer and target using point locations. You can then handle the `TargetVisibilityChanged` event to receive a notification each time the target becomes visible or obstructed from the observer.

To add a `LocationLineOfSight` analysis to your scene view:

1. Create map points to define the location of the observer and target in the line of sight analysis.

```
// location of the observer
MapPoint observerPoint = new MapPoint(-117.33, 33.56, 500.0, SpatialReferences.Wgs84);

// location of the target
MapPoint targetPoint = new MapPoint(-117.53, 33.76, 750.0, SpatialReferences.Wgs84);
```

2. Create a new `LocationLineOfSight`. In the constructor, provide the points that define the observer and target.

```
LocationLineOfSight locationLineOfSight = new LocationLineOfSight
                                   (observerPoint,
                                    targetPoint);
```

3. (**Optional**) Assign custom colors for the analysis results: visible and obstructed line segments. The colors you assign will be used for all line of sight analyses in the scene view.

```
Color visibleColor = Color.FromArgb(150, 0, 255, 127);
Color obstructedColor = Color.FromArgb(150, 255, 99, 71);

LineOfSight.VisibleColor = visibleColor;
LineOfSight.ObstructedColor = obstructedColor;
```

4. Add the analysis to an `AnalysisOverlay` to display the results (as described in Manage analyses with analysis overlays).
5. (**Optional**) Handle an event to know when target visibility changes.

```
locationLineOfSight.TargetVisibilityChanged += VisibilityChanged;
```

Location line of sight is generally used for evaluating visibility between a fixed observer and target, although those locations could be updated. Other factors in the scene view could also change the target visibility, such as adding or removing 3D features or updating surface data.

GeoElement line of sight

A more dynamic approach for evaluating line of sight is to use a `GeoElementLineOfSight` analysis. This analyzes visibility between point geoelements that represent the observer and target. As the position of these elements is updated (from a GPS feed, for example), the line of sight result will also be updated. As with the `LocationLineOfsight` analysis, you can handle the

Copyright © 1995-2020 Esri. All rights reserved.

`TargetVisibilityChanged` event to receive a notification each time the target becomes visible or obstructed. In the following image, for example, the target geoelement is selected when it becomes visible and unselected when it's obstructed.

Note:
Geo elements used to define an observer and target in a geo element line of sight analysis must have point geometry.

To add a `GeoElementLineOfSight` analysis to your scene view:

1. Create a new `GeoElementLineOfSight`. In the constructor, define the observer and target by passing in geo elements (with point geometry) that define their location.

```
// create a line of sight and define the observer and target with point GeoElements
GeoElementLineOfSight goodGuyBadGuyLOS = new GeoElementLineOfSight(friendlyGraphic,
hostileGraphic);
```

2. (**Optional**) Define offset values to apply to the observer and/or target locations (x, y, z). These offsets are defined in meters using a coordinate system relative to the center of the object, where X is to the right or left, Y is forward or back, and Z is up and down. To place an observer in the driver's seat of a car, for example, you might place her 60 centimeters from the left of the car center, 70 centimeters forward from center, and 40 centimeters above the center. These offset values would be (-0.6, 0.7, 0.4).

```
// apply some offsets (values are meters)
goodGuyBadGuyLOS.ObserverOffsetX = 1.0;
goodGuyBadGuyLOS.ObserverOffsetY = 2.0;
goodGuyBadGuyLOS.TargetOffsetX = 1.0;
goodGuyBadGuyLOS.TargetOffsetZ = 4.5;
```

3. (**Optional**) Assign custom colors for analysis results: visible and obstructed line segments. These are green and red by default.

```
Color visibleColor = Color.FromArgb(150, 0, 255, 127);
Color obstructedColor = Color.FromArgb(150, 255, 99, 71);

// these are static properties that apply to all lines of sight in the scene view
LineOfSight.VisibleColor = visibleColor;
LineOfSight.ObstructedColor = obstructedColor;
LineOfSight.LineWidth = 3.0;
```

Copyright © 1995-2020 Esri. All rights reserved.

4. Add the analysis to an `AnalysisOverlay` to display the results (as described in Manage analyses with analysis overlays).
5. (**Optional**) Handle an event to know when target visibility changes.

```
goodGuyBadGuyLOS.TargetVisibilityChanged += (s, e) =>
{
    // Enable weapons when target is visible
    if (goodGuyBadGuyLOS.TargetVisibility == LineOfSightTargetVisibility.Visible)
    {
        EnableDeathRay();
    }
};
```

Related topics

Manage analyses with analysis overlays
Make measurements

Copyright © 1995-2020 Esri. All rights reserved.

Trace a utility network

On the ArcGIS platform, utility networks offer a framework for modeling utility systems, such as electric, gas, water, storm water, wastewater, and telecommunications. Each utility network demonstrates how features are connected and how dynamic devices are configured. This topic describes how to build ArcGIS Runtime apps that can trace the flow of resources, such as gas, water, and electricity, through its network. For an overview of utility networks, see ArcGIS Pro's help topic What is a utility network?.

You can explore how a network is affected by real-world events such as storms, outages, or equipment failure, by asking questions like,

- How is your network connected?
- How does electricity/gas/water reach your house?
- If a device is disabled, what section of the network will be out of power?

To help answer these questions, ArcGIS Runtime supports the following utility trace types:

- **Upstream—**In a source-based network (gas or electric), an upstream trace is against the flow and toward the source, such as a circuit breaker or generator (subnetwork controller). In a sink-based network (sewer or storm water), an upstream trace is against the flow and away from the sink such as a sewage treatment (subnetwork controller). For more information, see ArcGIS Pro's help discussion on upstream traces
- **Downstream—**In a source-based network, a downstream trace is with the flow and away from the source, such as a circuit breaker or generator. In a sink-based network, a downstream trace is with the flow and toward the sink. For more information, see ArcGIS Pro's help discussion on downstream traces.
- **Subnetwork—**A trace that discovers all features participating in a subnetwork. This trace type is useful for validating whether subnetworks, such as circuits or zones, are defined or edited appropriately. The trace begins at one or more starting points and spans outward along connected features to find subnetwork controllers that are traversable. A subnetwork trace stops when it encounters a barrier, when it encounters a feature that is not traversable, or when there are no more connected features. For more information see ArcGIS Pro's help discussion on subnetwork trace.
- **Isolation—**A trace used to determine the minimum set of operable assets (point and line features) required to stop a network's resource from traveling/flowing, effectively isolating an area of the network. For instance, when a leak occurs on a water network, particular valves must be closed to eliminate water flow at the leak location. This prevents damage and allows field crews to safely start the repair process. For more information, see ArcGIS Pro's help discussion on locating isolating features.
- **Connected—**A trace that begins at one or more starting points and spans outward radially along connected features. A trace stops when a barrier is encountered or there are no more connected features. Can be used for validating newly edited features to ensure they are connected as expected. For more information, see ArcGIS Pro's help discussions on connectivity and finding connected features.

To trace a utility network you need to:

1. Use ArcGIS Pro to create a utility network
2. In ArcGIS Runtime, access the utility network
3. Define the trace parameters
   a. Decide which trace type to perform
   b. Define the starting and barrier locations
   c. Specify the trace configuration

Copyright © 1995-2020 Esri. All rights reserved.

4. Run the trace
5. Examine the results

Access the utility network

Utility networks are implemented in service-based geodatabases as network controller datasets. These datasets contain a network's service feature tables along with the network's domains, sources, tiers, assets, terminals, rules and associations. This utility network is accessible via these service feature tables stored in a single feature service.

You can display and share a complete utility network with a user via a web map if the map includes one feature layer for every network source.

To display and share the utility network, create a utility network object from a feature service URL and a web map that contains all the layers that participate in the utility network.

Note:
The feature service provides access to the topological network in the utility network. So, you could provide a map that contains just a subset of the feature layers, for a specific workflow. Any tracing would be performed using the full topological network provided by the feature service. If you need to add additional utility network layers you can create them from the individual network sources, as required.

You can also access a utility network and run a trace completely without a map. Just provide the feature service URL when you create the utility network. If needed, you can create a completely new map by creating feature layers from the network sources.

Note:
ArcGIS Runtime supports Utility Network version 2 and later that is provided from ArcGIS Pro 2.2 onwards. For details see utility network upgrade history.

Load the utility network

The utility network follows the loadable pattern for asynchronous resources. Loading the utility network loads the entire utility network schema (information about the datasets that participate in the network). Once loaded, your app can navigate this network schema to discover the domain networks it contains and any further details about the network.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create and load the utility network.
_utilityNetwork = await UtilityNetwork.CreateAsync(new Uri(FeatureServiceUrl), MyMapView.Map);
```

Define the trace parameters

Trace parameters define how the trace analysis proceeds across the utility network. These are the essential trace parameters:

1. Trace type
2. Start and barrier locations
3. Trace configuration

Trace type

ArcGIS Runtime supports the trace types described above (upstream, downstream, and so on).

Use the utility trace type to create the parameters

Create a set of `UtilityTraceParameters` by providing a `UtilityTraceType` of `Upstream`, `Downstream`, `Subnetwork`, `Isolation`, or `Connected`, along with a collection of starting locations (if known at this stage).

```
// Build trace parameters.
UtilityTraceParameters traceParams = new UtilityTraceParameters(UtilityTraceType.Downstream,
_startingLocations);
```

Start and barrier locations

Each trace requires one or more locations from which to initiate the trace. Optionally, you can also include barrier locations. Starting and barrier locations are defined using instances of `UtilityElement` which are added to the trace configuration's starting locations and barriers collection respectively.

You can create a starting location using steps like the following:

1. Create a `UtilityElement` using a feature.

   ```
   UtilityElement startingPointElement = _utilityNetwork.CreateElement(feature);
   ```

   a. If the feature represents a line, you can optionally specify a location along the line to use as the starting location. This value is expressed as a percentage of the length of the line, beginning from the line's from point. Set the `FractionAlongEdge` value to define the location of the point along the line.

      The following example uses the `GeometryEngine` to get the fraction of a tap location along the line.

      ```
      // Determine how far the starting point is located along a line
      Polyline line = GeometryEngine.RemoveZ(line) as Polyline;
      double fraction = GeometryEngine.FractionAlong(line, e.Location, -1);
      if (double.IsNaN(fraction)) { return; }
      utilityElement.FractionAlongEdge = fraction;
      ```

   b. If the feature represents a device with terminals, you must specify which terminal you want to use as the starting point.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Get the terminals for the feature.
IEnumerable<UtilityTerminal> terminals =
utilityElement.AssetType.TerminalConfiguration?.Terminals;
// Select the first terminal if more than one.
if (terminals?.Count() >= 1)
{
    utilityElement.Terminal = terminals.First();
}
```

2. Add the utility element to the utility trace parameters' `StartingLocations` collection.

```
traceParams.StartingLocations.Add(utilityElement);
```

3. If you need to add a barrier, complete step 1 above then add the utility element representing a barrier to the utility trace parameters' `Barriers` collection.

Trace configuration

If you don't want to use the trace configuration as created in ArcGIS Pro, you can either modify it or you can override it with settings you specify with ArcGIS Runtime. A trace configuration is set on the `UtilityTraceParameters`.

These settings allow you to do things like:

- Stop the trace at protective devices if they are open. For example, the flow of electricity in a network will be stopped if a fuse is open.
- Control the types of features traced. For example, trace only pipes with a diameter greater than six inches.

Each trace configuration manages basic properties such as:

- Include barriers in trace results
- Include containers in trace results
- Include content in trace results
- Include structures in trace results
- Add filter barriers (required for isolation traces)
- Ignore barriers if they are the starting points
- Domain network
- Source tier

For more advanced properties, such as traversability, propagators, and target tiers see the advanced trace configuration section.

When a utility network administrator creates a new tier in ArcGIS Pro, a subnetwork trace configuration is created and populated as described in Configure a trace.

You can choose if your app uses the trace configuration as defined by an administrator, a modified version of the configuration, or your own trace configuration.

Use a trace configuration defined in a utility network tier

To obtain the trace configuration from a utility network tier, you need to know the name of the domain network and the tier.

Copyright © 1995-2020 Esri. All rights reserved.

1. Obtain the utility network definition from the utility network.
2. Get the domain network from the utility network definition.
3. Obtain the tier from the domain network.
4. Pass the tier's trace configuration to the utility trace parameters.
5. Modify any of these properties as required.
   The following example illustrates using the preceding steps to get the configuration for a tier in the network.

```
// Set the trace configuration using the tier from the utility domain network.
UtilityDomainNetwork domainNetwork =
_utilityNetwork.Definition.GetDomainNetwork("ElectricDistribution");
UtilityTier mediumVoltageTier = domainNetwork.GetTier("Medium Voltage Radial");
UtilityTraceConfiguration mediumVoltageTierTraceConfig = mediumVoltageTier.TraceConfiguration;
traceParams.TraceConfiguration = mediumVoltageTierTraceConfig;
```

Create your own trace configuration

You can create your own trace configuration.

1. Create a utility trace configuration.
2. If you are running an upstream, downstream, or subnetwork trace then you must set the domain network as follows:
   a. Obtain the domain network from the utility network definition.
   b. Pass the domain network to the utility trace configuration.
3. Modify any of the other properties, as required
4. Pass the utility trace configuration to the utility trace parameters.

Execute the trace

Run the trace by calling the trace method on the utility network object. Use the utility trace parameters defined in the previous section. In this release of ArcGIS Runtime, all utility trace results are returned as utility element trace results.

```
// Execute the trace and get results.
IEnumerable<UtilityTraceResult> traceResult = await _utilityNetwork.TraceAsync(parameters);
UtilityElementTraceResult elementTraceResult = traceResult?.FirstOrDefault() as
UtilityElementTraceResult;

// Check if there are any elements in the result.
if (elementTraceResult?.Elements?.Count > 0)
{
    // TODO: process results.
}
```

If the trace fails you can examine why. For example, failure could be due to dirty areas in the network topology.

Examine the results

To display the results of this trace on a map, or to analyze them further, you need to obtain the corresponding features for these elements as described in the following steps.

1. Filter the utility element trace results to find those that are part of the map, and group them by network source name.

Copyright © 1995-2020 Esri. All rights reserved.

2. For every group (network source with utility elements), make sure there is a layer in the map for the features. Next find the features corresponding to the utility elements.
3. Select the features that correspond to the trace result or process as required.

```
// Check if there are any elements in the result.
if (elementTraceResult?.Elements?.Count > 0)
{
    // Get the result features from each network source layer in the map.
    foreach (FeatureLayer layer in MyMapView.Map.OperationalLayers.OfType<FeatureLayer>())
    {
        IEnumerable<UtilityElement> elements = elementTraceResult.Elements.Where(element =>
element.NetworkSource.Name == layer.FeatureTable.TableName);
        IEnumerable<Feature> features = await
_utilityNetwork.GetFeaturesForElementsAsync(elements);

        // Select features in this layer.
        layer.SelectFeatures(features);
    }
}
```

Advanced trace configuration

Traversability

As you trace a topological network you can examine a number of constraints or conditions that could stop the trace. For example, you can stop tracing the network if:

- A water valve is closed
- An electric switch is open
- A distance of 1000 m is reached
- The gas pipe is made of plastic

The ability for a trace to traverse the topological network is defined by the `Traversability` property on the `UtilityTraceConfiguration` class. You can set conditions or constraints to this trace using barriers and function barrier properties.

Barriers and function barriers used as traversability constraints are defined using a condition, such as a valve being closed, encountering a particular material type, reaching a specified threshold along the trace, and so on. This contrasts with barriers that may be defined with trace parameters, which are defined using utility network elements, often selected interactively by the user.

Barriers

Set up a trace barrier by comparing the value of an asset's attribute or by examining the existence of a `UtilityCategory`. You can compare them individually or combine them with boolean And / Or operations into complex filters.

- Use the `UtilityNetworkAttributeComparison` class to compare an element's network attribute. For example, compare an attribute value to:
  - A specific value (for example, "DeviceStatus not equal to 4"), and/or
  - Another network attribute on the same element (for example, "SeasonalDeviceStatus" <> "NormalDeviceStatus")

```
// Get the "Lifecycle" network attribute
UtilityNetworkAttribute lifecycleNetworkAttribute =
```

Copyright © 1995-2020 Esri. All rights reserved.

```
_utilityNetwork.Definition.GetNetworkAttribute("Lifecycle");
if(lifecycleNetworkAttribute == null) { return; }

// Create a network attribute comparison that stops traversal if Lifecycle <> 4)
UtilityNetworkAttributeComparison inDesignNetworkAttributeComparison = new
UtilityNetworkAttributeComparison(lifecycleNetworkAttribute,
UtilityAttributeComparisonOperator.NotEqual, 4);

// Create a network attribute comparison to stop traversal if Lifecycle <> 8
UtilityNetworkAttributeComparison inServiceNetworkAttributeComparison = new
UtilityNetworkAttributeComparison(lifecycleNetworkAttribute,
UtilityAttributeComparisonOperator.NotEqual, 8);

// Combine these two comparisons together with "And"
UtilityTraceAndCondition lifecycleFilter = new
UtilityTraceAndCondition(inDesignNetworkAttributeComparison,
inServiceNetworkAttributeComparison);

// Final condition stops traversal if Lifecycle <> 4 and Lifecycle <> 8
traceConfiguration.Traversability.Barriers = lifecycleFilter ?? null;
```

- Check a utility element's asset type to see whether that asset type (and thus the element) is included in a specific category using the `UtilityCategoryComparison` class.

Function barriers

You can create a function barrier to terminate network traversal whenever a function expression evaluates to true. The function barrier compares the current results of a function and a given value. For example, you can use the function barrier to stop traversal after the trace traverses 1000 m along the network.

```
// Create a network attribute object for the Shape length network attribute from the utility
network definition
UtilityNetworkAttribute shapeLengthNetworkAttribute =
_utilityNetwork.Definition.GetNetworkAttribute("Shape length");
if (shapeLengthNetworkAttribute == null) { return; }

// Create a function that adds up shape length
UtilityTraceFunction lengthFunction = new UtilityTraceFunction(UtilityTraceFunctionType.Add,
shapeLengthNetworkAttribute);

// Create a function barrier that stops traversal after 1000 meters
UtilityTraceFunctionBarrier distanceBarrier = new UtilityTraceFunctionBarrier(lengthFunction,
UtilityAttributeComparisonOperator.GreaterThan, 1000.0);

// Set this function barrier
traceConfiguration.Traversability.FunctionBarriers.Add(distanceBarrier);
```

For more information see traversability.

Note:
The `UtilityTraversabilityScope` property determines whether these conditions are evaluated on edges, junctions, or both.

Trace filters

Filters are a mechanism to stop tracing when returning results. They do not stop traversability to the controller.

Copyright © 1995-2020 Esri. All rights reserved.

A trace filter and traversability both have properties for defining barriers, function barriers, and scope. These properties work the same way in both classes. What makes them different is when they get applied and how they affect the trace.

While both traversability and trace filter can terminate a trace, they have slightly different use cases. In upstream and downstream traces, traversability is considered first because it determines the subnetwork controller and the flow direction of tracing. If the controller's type is source-based, the flow direction is away from the identified subnetwork controller. If the controller's type is sink-based, the flow direction is toward the controller. Once the starting location, subnetwork controller, and flow direction are all established, the features that are traversable are then evaluated against the trace filter criteria.

If you want a trace to find the next upstream protective device in an electrical network, for example, you would create a `UtilityCategoryComparison` where 'Protective Device' category exists. If you set this barrier on traversability, the trace will fail. It will be unable to find a subnetwork controller to determine which direction is upstream. You should use a trace filter instead.

Trace filter barriers can be used to configure isolation traces. An isolation trace allows you to isolate a portion of the network using filter barriers. An upstream trace configured to use isolation valves as filter barriers will determine the boundary of the isolation zone, from which you can determine which customers are affected (out of service).

```
// Define a starting location utility element
UtilityNetworkSource networkSource = utilityNetwork.Definition.GetNetworkSource("Gas Line");
UtilityAssetGroup assetGroup = networkSource.GetAssetGroup("Distribution Pipe");
UtilityAssetType assetType = assetGroup.GetAssetType("Bare Steel");
UtilityElement startingLocation = utilityNetwork.CreateElement(assetType, new
Guid("{B903103D-8933-4383-8C7B-A77FC705F32D}"));

// Create trace parameters with 'isolation' trace type and the starting location
var parameters = new UtilityTraceParameters(UtilityTraceType.Isolation, new[] { startingLocation
});

// Create a category comparison ("IsolationDevice")
UtilityCategory category = utilityNetwork.Definition.Categories.Single(c => c.Name ==
"IsolationDevice");
var categoryComparison = new UtilityCategoryComparison(category,
UtilityCategoryComparisonOperator.Exists);
// Create a network attribute comparison ("Accessible == 1")
UtilityNetworkAttribute networkAttribute =
utilityNetwork.Definition.GetNetworkAttribute("Accessible");
var networkAttributeComparison = new UtilityNetworkAttributeComparison(networkAttribute,
UtilityAttributeComparisonOperator.Equal, 1);
// Create a trace condition that uses both comparisons defined above
var condition = new UtilityTraceAndCondition(categoryComparison, networkAttributeComparison);

// Create a trace configuration and assign it to the trace parameters
parameters.TraceConfiguration = new UtilityTraceConfiguration
{
    // Create a trace filter that uses the condition
    Filter = new UtilityTraceFilter
    {
        Barriers = condition
    },
    IncludeIsolatedFeatures = true
};
// Execute the trace with these parameters
IEnumerable<UtilityTraceResult> result = await utilityNetwork.TraceAsync(parameters);
```

Copyright © 1995-2020 Esri. All rights reserved.

Bitset network attributes

Bitset network attributes are only applicable to upstream and downstream trace types. They can be used to add special logic during a trace so the trace is more reflective of real world scenarios.

There are cases where traces need to be aware that a network attribute is a bitset that controls traversability. For example, you might have an electrical network in which phase is represented as a bitset network attribute (one bit per phase), and overhead electrical devices are represented with one device per phase. You could use a bitset network attribute to ensure the trace results include valid paths that are specified in the network attribute, not all paths.

Nearest neighbor

The nearest neighbor filter, `UtilityNearestNeighbor`, allows you to return a specified number of features from the starting location of the trace. When assigned to a `UtilityTraceFilter`, it will return a number of features of a certain type within a given distance.

A network attribute that represents travel cost is used to define the distance, which is typically shape length. Other attributes may be more useful depending on circumstances. For example, if you are searching for the "nearest" vault in an underground structure network, you may prefer a geographically distant vault that is connected via a duct bank rather than a closer one through a direct-buried trench (since excavating the trench is more costly). In this case a different attribute that represents the cost per segment should be used.

The type of features to be returned can be specified by utility category, asset type, or both. A valid `UtilityNearestNeighbor` therefore, will have a cost network attribute, a count greater than 0, and at least one specified category or asset type.

Propagators

A propagator defines the propagation of a network attribute along a traversal and provides a filter to stop traversal. Propagators are only applicable to subnetwork-based traces (subnetwork, upstream, or downstream). One example is electric phase propagation, where open devices along the network will restrict some phases from continuing along the trace.

```
// Get a network attribute object for the "Phases Normal" attribute from the utility network
definition
UtilityNetworkAttribute phasesNormalNetworkAttribute =
_utilityNetwork.Definition.GetNetworkAttribute("Phases Normal");
if (phasesNormalNetworkAttribute == null) { return; }

// Create a propagator to propagate the attribute using a Bitwise And function
UtilityPropagator propagator = new UtilityPropagator(phasesNormalNetworkAttribute,
UtilityPropagatorFunctionType.BitwiseAnd, UtilityAttributeComparisonOperator.IncludesAny, 7);

// Assign the propagator to the trace configuration
traceConfiguration.Propagators.Add(propagator);
```

For more information see propagators.

Target tiers

All upstream and downstream traces can operate across the current tier (source tier). If you want your upstream or downstream trace to continue into another tier, you set the `TargetTier` property on the `UtilityTraceConfiguration`.

Copyright © 1995-2020 Esri. All rights reserved.

Tracing considerations

Dirty areas

All tracing operations rely on a topological index that is built from the utility network. If the network has been edited but the topology is out of date then it can contain dirty areas. For more information see ArcGIS Pro's discussion validate the network topology. If the topological network has dirty areas you can adopt a different approach depending on your app's purpose:

- If the app must trace with the latest data (for example, an outage management app), an error should be returned to the user if it encounters a dirty area.
- If the app can trace with a network that is out of sync with the feature (for example, a pole inspection app), then you should consider whether to set `ValidateConsistency` to `False` on the `UtilityTraceConfiguration`. You can optionally display a warning to the user if a dirty area is encountered.

Get associated utility elements

Associations model the following types of relationships between two utility network elements:

Association	Description	Geometry supported
Connectivity	Models the connectivity between two junctions that don't have geometric coincidence (are not in the same x, y and z location). A transformer may be connected to a fuse, for example.	Yes
Structural attachment	Models equipment attached to structures. A transformer bank may be attached to a pole, for example.	Yes
Containment	Models assets that contain other assets. A vault may contain valves and pipes, for example.	No

An association is defined between two `UtilityElement` objects. You can identify which `UtilityElements` are associated with a given `UtilityElement` using one of the `GetAssociationsAsync` methods on the `UtilityNetwork`.

```
// Get the elements associated (by containment) with a specified utility element
IEnumerable<UtilityAssociation> associations = await
_utilityNetwork.GetAssociationsAsync(identifiedElement, UtilityAssociationType.Containment);
if(associations == null) { return; }

// Iterate all associations for this element
foreach(UtilityAssociation association in associations)
{
    // Get the first ("from") and second ("to") elements in the association
    UtilityElement fromElement = association.FromElement;
    UtilityElement toElement = association.ToElement;
}
```

If the association represents a connectivity or structural attachment association, it may include a geometry value (polyline) representing the connection relationship between a **from element** and a **to element**. You can use the geometry to visualize the association as a graphic in the map.

If you want to find all the valid associations within a specific extent, call the `GetAssociationsAsync` method and provide an envelope that defines the specific extent.

Copyright © 1995-2020 Esri. All rights reserved.

The following example creates a new graphics overlay to display all associations within the map extent using a unique symbol for each `UtilityAssociationType`.

```
// Create a graphics overlay for associations.
GraphicsOverlay associationsOverlay = new GraphicsOverlay();
MyMapView.GraphicsOverlays.Add(associationsOverlay);

// Create symbols for structural attachment and connectivity associations.
Symbol attachmentSymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Dot, Color.Green, 5d);
Symbol connectivitySymbol = new SimpleLineSymbol(SimpleLineSymbolStyle.Dot, Color.Red, 5d);

// Create a unique value renderer for the associations and apply it to the graphics overlay.
var attachmentValue = new UniqueValue("Attachment", string.Empty, attachmentSymbol,
UtilityAssociationType.Attachment.ToString());
var connectivityValue = new UniqueValue("Connectivity", string.Empty, connectivitySymbol,
UtilityAssociationType.Connectivity.ToString());
associationsOverlay.Renderer = new UniqueValueRenderer(new List<string> { "AssociationType" },
new List<UniqueValue> { attachmentValue, connectivityValue }, string.Empty, null);

// Get the current viewpoint's extent (exit if null).
Envelope extent =
MyMapView.GetCurrentViewpoint(ViewpointType.BoundingGeometry)?.TargetGeometry?.Extent;
if (extent == null)
{
    return;
}

// Get all of the associations in the current extent.
IEnumerable<UtilityAssociation> associations = await
_utilityNetwork.GetAssociationsAsync(extent);
foreach (UtilityAssociation association in associations)
{
    // If it's not a containment relationship, add a graphic for the association.
    if (association.AssociationType != UtilityAssociationType.Containment)
    {
        Graphic graphic = new Graphic(association.Geometry);
        graphic.Attributes["GlobalId"] = association.GlobalId;
        graphic.Attributes["AssociationType"] = association.AssociationType.ToString();
        associationsOverlay.Graphics.Add(graphic);
    }
}
```

Copyright © 1995-2020 Esri. All rights reserved.

For more information see ArcGIS Pro's help topic Associations.

Copyright © 1995-2020 Esri. All rights reserved.

Use the cloud and servers

Copyright © 1995-2020 Esri. All rights reserved.

Access the ArcGIS platform

The ArcGIS platform provides a full range of GIS capabilities. There are various ways to add powerful functionality and intelligence to your ArcGIS Runtime app by connecting to platform capabilities.

ArcGIS Online

ArcGIS Online is a cloud-based, collaborative content management system for maps, apps, data, and other geographic content. You can access ArcGIS Online through mobile devices, a website (www.ArcGIS.com), and desktop map viewers. With ArcGIS Online you can do the following:

- Create web maps.
- Web enable your data.
- Share your maps, data, and applications.
- Find relevant and useful basemaps, data, and configurable GIS resources.
- Manage content and users in your organization.

Tip:

With your free ArcGIS for Developers account, you have access to ArcGIS Online for development and testing. Sign in at developers.arcgis.com or www.ArcGIS.com. You may want to upgrade to a paid ArcGIS Developer Subscription.

Using ArcGIS Runtime SDK, you can work with content in a portal, for example you can access and edit existing content and create new content items, search for users and groups, and share and unshare items.

Learn more about developing with ArcGIS Online in ArcGIS Online Help.

ArcGIS for organizations

To access all the capabilities of ArcGIS Online, your organization can purchase a subscription. With this subscription, you can store and process geographic data, solve complex routing problems, perform spatial analysis, create reports, and enrich your own data with demographic and geographic attributes. You also gain access to various productivity apps. A subscription can help you make geospatial information more pervasive both within your organization and throughout the general public.

Tip:

Your free ArcGIS for Developers account includes some subscription-only services for development and testing. Sign in at developers.arcgis.com or www.ArcGIS.com. If you upgrade your free account to a paid plan, you can still sign in using both of these sites.

Learn more about ArcGIS for organizations at ArcGIS Online.

Portals, users, roles, groups, and sharing

ArcGIS Online is an information portal and is represented in the API by the `ArcGISPortal` class. This class is loadable.

When you sign in to ArcGIS Online with an organization account, you see a specialized view that your organization administrator has configured for you, giving you access to maps, content, and other services specific to your organization. You can also access all your own content and services.

Copyright © 1995-2020 Esri. All rights reserved.

A registered user of a portal is represented in the API by the `PortalUser` class. When you have signed in to a portal, you can get authentication information relating to the authenticated user from the portal class. Several options are available for signing in to a portal, such as OAuth 2.0, network credentials, tokens, and public keys (PKI). Two authentication patterns are available—named user and app login. For an overview of the ways to access secure services, see ArcGIS Security and Authentication.

Portals and named users have an essential role to play in some app licensing scenarios. For more information about licensing, see License your app.

Groups are a way to collaborate with other users who share a common interest. A user can create groups, join groups, and share items with those groups. Groups have titles, descriptions, thumbnails, and unique IDs to help users identify them. The sharing model within ArcGIS Online is based on groups. Groups are represented in the API by the `PortalGroup` class.

A free, public account is another way to access ArcGIS Online. These accounts are not associated with an organization and offer a limited set of functionality. A public account allows you to use and create maps and share your maps and apps with everyone. Public accounts are for noncommercial use only.

Connect to public content and services on ArcGIS Online

To connect to ArcGIS Online and access public content anonymously, you can begin by creating a portal, without authenticating with the portal.

```
// call the static method CreateAsync without arguments to get a reference to the arcgis.com
portal
ArcGISPortal portal = await ArcGISPortal.CreateAsync();

// get portal info and display it to the user
PortalInfo portalInfo = portal.PortalInfo;

Console.WriteLine(string.Format("Access: {0}", portalInfo.Access.ToString()));
Console.WriteLine(string.Format("Name: {0}", portalInfo.PortalName));
Console.WriteLine(string.Format("Mode: {0}", portalInfo.PortalMode.ToString()));
```

From here, you can access public content; for example, you can display a web map, or access the data stored in a portal item. You can also search for content such as web maps, map services, map features, groups, and users.

Some organizations share content publicly and allow anonymous access to that content; connect to publicly accessible content from such organizations by specifying the organization URL. For example:

```
// call the static method CreateAsync with a Uri argument to get a reference to a specific portal
ArcGISPortal portal = await ArcGISPortal.CreateAsync(new
Uri("http://anorganization.maps.arcgis.com/sharing/rest"));
```

Connect to secured content and services on ArcGIS Online

Apps that target organization users who are known to the ArcGIS platform (named users) should pass a credential for the user when creating the portal object.

```
// generate an ArcGISTokenCredential using input from the user (username and password)
var cred = await AuthenticationManager.Current.GenerateCredentialAsync(
                                                 new Uri("http://anorganization.maps.arcgis.com/
sharing/rest"),
                                                 userName,
```

Copyright © 1995-2020 Esri. All rights reserved.

```
                                                    password) as ArcGISTokenCredential;

// connect to the portal, pass in the token
ArcGISPortal portal = await ArcGISPortal.CreateAsync(
                                                    new Uri("http://anorganization.maps.arcgis.com/
sharing/rest"),
                                                    cred,
                                                    CancellationToken.None);

// get the current portal user and check privileges
PortalUser user = portal.User;
IEnumerable<PortalPrivilege> privileges = user.Privileges;
```

The portal object now has access to all the secure content for which the user has access rights and can be used to find out more information about the user, such as the user's full name (instead of the account user name). Additionally, information about the organization such as the name, banner image, description, and so on, can be found as shown above. Apps often make use of this information when a user connects to a specific portal, to show the user organization branding and context.

Typically, the portal object with the authenticated user is cached and used throughout the app session, to provide the app with a view of a portal that is centered around a single user. When the app is restarted, the credential must be reinstated, or the user must repeat the authentication process.

To access secure content, an app needs to provide a way to sign in to the platform, a process often known as authentication. The recommended approach for authenticating a user known to the platform is to use a user login and OAuth. Apps that target users who are unknown to the ArcGIS platform can authenticate with the platform on behalf of the user by using an app login.

ArcGIS Enterprise

ArcGIS Enterprise provides you with the same core capabilities as ArcGIS Online, but it can be installed and hosted on your own premises, behind your firewall, for controlled distribution of content.

Learn more about ArcGIS Enterprise.

Connect to ArcGIS Enterprise portal

Connecting to an instance of ArcGIS Enterprise portal is done in a very similar way to connecting to ArcGIS Online and is represented in the API by the same class, `ArcGISPortal`. Use the URL to the Enterprise portal website, along with an appropriate credential valid on that portal, or no credential if accessing public content anonymously.

```
// generate an ArcGISTokenCredential using input from the user (username and password)
var cred = await AuthenticationManager.Current.GenerateCredentialAsync(
                                                    new Uri("http://geoportal.mycompany.com/sharing/
rest"),
                                                    userName,
                                                    password) as ArcGISTokenCredential;

// connect to the portal, pass in the token
ArcGISPortal portal = await ArcGISPortal.CreateAsync(
                                                    new Uri("http://geoportal.mycompany.com/sharing/
rest"),
                                                    cred,
                                                    CancellationToken.None);

// get the current portal user and check privileges
PortalUser user = portal.User;
IEnumerable<PortalPrivilege> privileges = user.Privileges;
```

Copyright © 1995-2020 Esri. All rights reserved.

ArcGIS Server

ArcGIS Server (a component of ArcGIS Enterprise) is a complete and integrated server-based GIS, allowing you to publish many different types of GIS services for use by a variety of clients including ArcGIS Runtime apps. Some services provide similar functionality to that available from ArcGIS Online—for example, you can create map layers based on map, feature, or WMS services, and perform geocoding, routing, or geoprocessing functions by using services hosted by ArcGIS Server.

Discover content and services on ArcGIS Server

Use the ArcGIS Services Directory to browse the available services on an ArcGIS Server site. Open the REST Services home page in a browser, for example, http://sampleserver6.arcgisonline.com/arcgis/rest/services.

Contact your ArcGIS Server system administrator if you're uncertain about the server URL or ArcGIS instance.

Connect to ArcGIS Server

Connecting to ArcGIS Server is a little different from connecting to portals—in the place of connecting to the portal itself, you pass the URL of the service you want to use to a class designed to work with that service type.

For example, you can create a map layer by passing in the URL to a dynamic map service:

```
var uri = new Uri(@"http://sampleserver6.arcgisonline.com/arcgis/rest/services/DamageAssessment/
MapServer");
ArcGISMapImageLayer dynamicLayer = new ArcGISMapImageLayer(uri);
```

If the service is secured, valid credentials or authentication will be required.

ArcGIS Server services are listed below, along with the classes that make use of them.

Service type	API classes
Feature service	`FeatureLayer, FeatureTable`
Map service—Dynamic	`ArcGISMapImageLayer`
Map service—Tiled	`ArcGISTiledLayer`

ArcGIS Server services and related API classes

Learn more about ArcGIS for Server at esri.com.

Authentication Manager

Your app may need to access secured resources that are restricted to authorized users. For example, your organization may host private data layers or feature services that are only accessible by verified users. You may also need to take advantage of premium ArcGIS Online services, such as routing, that require secured user access. The ArcGIS platform provides many ways to secure access to your organization's content and services. The ArcGIS Runtime SDK provides full support for access to secured ArcGIS Server, ArcGIS Online, or ArcGIS Enterprise resources using the following authorization methods:

- ArcGIS Tokens: proprietary token-based authentication mechanism.
- OAuth 2.0: secure delegated access to server resources.
- Web-tier security: HTTP secured service / Integrated Windows Authentication (IWA).
- Certificate: Public Key Infrastructure (PKI).

Copyright © 1995-2020 Esri. All rights reserved.

Implementing these security methods in your app (potentially for a variety of platforms) can be a lot of work. To simplify authentication, ArcGIS Runtime provides classes to automate the process: Authentication Manager (`AuthenticationManager`) and Authentication Challenge (`ChallengeHandler`).

The Authentication Manager manages access by the app to secured resources. This singleton creates an Authentication Challenge whenever an authentication or security issue is encountered anywhere in the API. The types of Authentication Challenge include the following:

- Username / password: Challenges needing username / password authentication.
- OAuth: Challenges needing an OAuth authorization code.
- Client Certificate: Challenges needing a client certificate to be provided.
- Secure Sockets Layer (SSL) Handshake - Challenges needing a response to certain `SslError` errors, usually an untrusted host due to a self-signed certificate.

Your app responds to an Authentication Challenge by providing the requested authentication information, which may be a certificate, credentials, or username and password.

Note:

Because OAuth authorization requires additional information about the server, you must register servers using OAuth authorization with the authentication manager. In OAuth, client ID and client secret are created when an app is registered with a server. The client ID is considered public information, and represents a public identifier for an app. The client secret is a secret known only to the application and the authorization server, and must be kept confidential. The following types of OAuth authorization methods (grants) are available:

- Implicit: registration requires a client ID and a redirect URL. The user enters a login with the server and receives an access token.
- Authorization code: registration requires a client ID, client secret, and a redirect URL. The user enters a login with the server and receives an access token and refresh token.
- Client credentials: registration requires a client ID, client secret, and a redirect URL. The user does not need to enter a login, but rather is granted access on behalf of the app.

After your app provides the authentication information, the Authentication Manager caches that information. The Authentication Manager contains an instance of a `CredentialCache` which maintains a cache of credentials, in memory, that have been previously used to satisfy authentication challenges. This allows a credential to be reused where appropriate, and prevents unnecessary or duplicate challenges from being issued while accessing secure resources from the same security realm. Caching happens automatically if credential caching is enabled on the Authentication Manager. When the app wants to sign out, clear the credential cache within the Authentication Manager.

Related topics

Access portal content
Search for content
Share a portal item
Add items to a portal

Copyright © 1995-2020 Esri. All rights reserved.

Access portal content

Portals such as ArcGIS Online and ArcGIS Enterprise can store many types of content. Perhaps the type most commonly used by apps is a web map—a JSON description of the data in a map, along with other display and behavioral properties. Other types include tile packages, feature collections, and services. Some types, such as globe documents and map templates, may only be relevant to desktop systems. Although you can access these types of data using the API, you may not be able to use them on a mobile device. See Work with offline layers for more information.

`PortalItem` represents an individual item stored on a portal. Each item includes the following:

- Metadata, such as a unique identifier (ID), title, summary, description, thumbnail image, and tags
- Binary or text-based data—for example, the JSON that defines a web map or a feature collection, or a URL pointing to a map service, or the binary data of a tile package or shapefile

Apps use portal items in different ways. They may show a user a range of items from a portal by combining the title, thumbnail image, and other metadata. This approach is often used to allow users to browse web maps, basemaps, or map services, and identify the item to work with. If a user selects an item, the app may then download the data stored in the portal item, for example, by opening and displaying a web map in a map, or downloading a tile package. Apps sometimes access items directly by ID if a specific item should always be used.

Your app can also add new portal items and alter existing items by adding comments and ratings, updating item information, sharing, unsharing, and deleting them, as long as the authenticated user has access permissions to do so.

You access the content of a portal item differently depending on its type. Below you can find examples of accessing different types of portal items. For more information on connecting to a portal, see Access the ArcGIS platform.

Display a web map by ID

A very common task is to display a web map from a portal. Web maps are stored as portal items and can be opened by using the ID of the item, along with the URL of the portal. If the map is not public (for example, if it is shared only with a specific group), you also need to pass in valid credentials with permission to view the map. See Access the ArcGIS platform for more information on accessing secured items.

In the following example, the web map portal item is opened using the `PortalItem.CreateAsync` static method. The item is passed to the constructor for a new `Map`, which is then displayed in the map view.

Copyright © 1995-2020 Esri. All rights reserved.

```
// connect to ArcGIS Online (default portal if no URL is passed to ArcGISPortal.CreateAsync)
ArcGISPortal arcGISOnline = await ArcGISPortal.CreateAsync();

// create an PortalItem using a portal item ID
var portalItem = await PortalItem.CreateAsync(arcGISOnline, "01f052c8995e4b9e889d73c3e210ebe3");
// pass the portal item to the constructor for a new Map
var webMap = new Map(portalItem);

// add the map to the map view
MyMapView.Map = webMap;
```

There are different ways to display a map from a web map.

Open a map

You can create a map from a web map portal item without needing to display it immediately. From this, you could find out more about the contents of the map before displaying it if required—for example, getting the basemap, operational layers, bookmarks, and initial extent. To do this, load the map before setting it into a map view—see Loadable pattern for asynchronous resources for more information.

The following example gets a web map portal item, finds one of its bookmarks, and then zooms to the extent of the bookmark.

```
// connect to ArcGIS Online (default portal if no URL is passed to ArcGISPortal.CreateAsync)
ArcGISPortal arcGISOnline = await ArcGISPortal.CreateAsync();

// create an PortalItem using a portal item ID
var portalItem = await PortalItem.CreateAsync(arcGISOnline, "01f052c8995e4b9e889d73c3e210ebe3");
  // pass the portal item to the constructor for a new Map
var webMap = new Map(portalItem);

// load the map before accessing its contents
await webMap.LoadAsync();

// get a bookmark from the web map
var bookmark = webMap.Bookmarks.Where(b => b.Name == "Chula Vista").FirstOrDefault();
// if the bookmark was found, zoom the map view to it
if (bookmark != null)
{
    await MyMapView.SetViewpointAsync(bookmark.Viewpoint);
}
```

Note:

Remember that you must also specify credentials for a valid user with access to the item, if the item is not publicly shared.

Create a layer from a service item by ID

Portal items representing map or feature services can be used to create layers by using the portal item URL. The following example gets a portal item that represents a feature service and reads the URL. The first layer in the service is added to the map.

```
// connect to ArcGIS Online
ArcGISPortal arcGISOnline = await ArcGISPortal.CreateAsync();

// create an PortalItem using a portal item ID
var portalItem = await PortalItem.CreateAsync(arcGISOnline, "9fb4d96077e24bcba72ba6cf1139c9cf");
```

Copyright © 1995-2020 Esri. All rights reserved.

```
  // check the item type to make sure its a feature service
if(portalItem.Type != PortalItemType.FeatureService) { return; }

// build a URL to the first layer in the service
var uri = new Uri(portalItem.Url + "/0");
// create a new service feature table referencing the service
var table = new ServiceFeatureTable(uri);
// create a new feature layer from the table
var layer = new FeatureLayer(table);

// add the layer to the map
MyMapView.Map.OperationalLayers.Add(layer);
```

Find information about any portal item by ID

You can use the ID of any item stored in a portal to access it, and also to find its fields—properties such as its title, description, thumbnail image, owner, and any ratings or comments added by portal users. You can also find out the type of data it contains.

```
// connect to ArcGIS Online
ArcGISPortal arcGISOnline = await ArcGISPortal.CreateAsync();

// create an PortalItem using a portal item ID
var portalItem = await PortalItem.CreateAsync(arcGISOnline, "9fb4d96077e24bcba72ba6cf1139c9cf");

// get some information about the item
var title = portalItem.Title;
var createdOn = portalItem.Created;
var numViews = portalItem.ViewCount;

var itemInfo = string.Format("{0} was created on {1} and has been viewed {2} times.", title,
createdOn, numViews);
```

Note:

The thumbnail of a portal item, along with the ratings and data, is not returned from the portal when you initially load the item, in order to reduce the network traffic and memory requirements. See the section below on how to access these properties.

Fetch thumbnails of items

When you create a portal item object, not all information associated with it is immediately returned. This allows you to work with the item using a minimum amount of memory and delay fetching the larger pieces of data unless, or until, you really need them. Information you need to fetch, if required, includes thumbnail images, ratings and comments, item data, group memberships, and user folder contents.

The following example shows how to create a bitmap image to display a portal item thumbnail.

```
// get the URL to the portal item thumbnail image
var thumbUri = portalItem.ThumbnailUri;
// create a new bitmap using the URL
var thumbImage = new BitmapImage(thumbUri);
// display the image in a control in the app
ItemThumbnailImage.Source = thumbImage;
```

Copyright © 1995-2020 Esri. All rights reserved.

The same applies to the thumbnails of other classes, like users, groups, and organizations. Use `ThumbnailUri` on `PortalItem`, `PortalGroup`, and `PortalUser`. Additionally, `PortalInfo` provides `ThumbnailUri` (for the organization) and `PortalThumbnailUri`.

Access item data

Portal items that represent files, such as tile packages, images, documents, PDF files, and so on, can be opened, downloaded, and then stored or opened in your app or in another app. As in the previous example, you will need to specifically fetch the data of a portal item after you create it. Use the `GetDataAsync` method to do this.

Fetch a user's content

Many apps present a list of maps or other items that belong to the current user. Users can create folders within their portal accounts to help organize their work. Apps should respect the user's view of their content and present the list of folders to the user, allowing the user to choose which folder to look at in detail.

It's easy to get all the portal items owned by the authenticated user (that represent the content and services items created by that user). `PortalUser` represents information about a user, and from this you can retrieve all of the items and folders (represented by `PortalFolder`) owned by the user.

The following example gets folders and items belonging to an authenticated user of a portal.

```
// get the current user (will be null if connected anonymously)
var thisUser = myPortal.User;
if(thisUser == null) { return; }
  // get folders and items for the user
var usersContent = await thisUser.GetContentAsync();
var folders = usersContent.Folders;
var portalItems = usersContent.Items;

// report the title of each portal item
Debug.Print("Items for " + thisUser.FullName + ":");
foreach(var itm in portalItems)
{
    Debug.Print(itm.Title);
}
  // report the contents of the user's folders
foreach(var f in folders)
{
    Debug.Print("Contents of folder '" + f.Title + "'");
    foreach (var i in await thisUser.GetContentAsync(f.FolderId))
    {
        Debug.Print("--" + i.Title);
    }
}
```

To get items shared with the current user, or items owned by a different user, you need to search for content.

List the groups a user belongs to

Groups can be configured in different ways—some groups are open to all users; some are only accessible by invitation; and for others, any user can request to join. The groups to which the authenticated user belongs can be accessed, and for any group, the identities of the members can be found.

Copyright © 1995-2020 Esri. All rights reserved.

The `PortalGroup` class represents a group. You can get a list of the groups a user belongs to by calling the `PortalUser.Groups` property. Use `GetGroupUsersAsync` to retrieve the users in each group—after that, you can get the members from the `Users` property.

```
var groups = thisUser.Groups;
foreach(var grp in groups)
{
    var groupMembers = await grp.GetGroupUsersAsync();
    var numUsers = groupMembers.Users.Count();
    var numAdmin = groupMembers.Admins.Count();
          string role = groupMembers.Admins.Contains(thisUser.UserName) ? "admin" :
"user";

    Debug.Print("Your role in " + grp.Title + " is " + role);
    Debug.Print("It has " + numUsers.ToString() + " users and " + numAdmin.ToString() + "
admins.");
}
```

From the group, you can find out if membership is by invitation only by examining the `IsInvitationOnly` property, and if the user can share items with the group using the `IsViewOnly` property.

Related topics

Access the ArcGIS platform
Search for content
Share a portal item
Add items to a portal

Copyright © 1995-2020 Esri. All rights reserved.

Search for content

Portals can contain a huge amount of information. To help your users find the information they want to work with, you can add searching to your app. Searching can allow users to find information in different ways, for example, by finding groups based on a keyword, or finding all portal items that are web maps with specific tags.

Portal administrators can configure a number of settings to highlight specific content to users, such as the set of available basemaps,the content featured in the portal homepage and gallery, and the featured groups for the organization. Learn how to easily find featured basemaps, content, and groups.

Searching is done from the `ArcGISPortal` class. Publicly accessible items and groups can be accessed without any authentication, but to find items that are secured, a user with access to the item must be authenticated with the portal. See Connect to secured content and services on ArcGIS Online for more information about authenticating with a portal.

Search by defining query parameters

The API provides a flexible, generic search, along with some convenience methods to help you perform common portal searches.

Find portal items by keyword

To search for portal items using a keyword, construct a query that includes the keyword you want to search for. By default, the query will return portal items that include the keyword in any of the common fields of a portal item, including the title, tags, description, and type. The results of the query can be iterated.

Create a `PortalQueryParameters` object and pass the search term (string) into the constructor. Then use the `PortalQueryParameters` in a call to the `FindItemsAsync` method. The resulting `PortalQueryResultSet<PortalItem>` contains portal items that use the search term in any of the searched fields.

Tip:

Use `PortalQueryParameters` with an `Envelope` to find portal items that intersect a specified extent. You can optionally provide a query string to define search parameters that include both spatial and attribute criteria.

```
var portal = await ArcGISPortal.CreateAsync();
// Create a new SearchParameters (or SpatialSearchParameters)
var searchParams = new PortalQueryParameters(searchText);
searchParams.Limit = 25;
// Search the portal for items matching the search criteria
PortalQueryResultSet<PortalItem> searchResultInfo = await portal.FindItemsAsync(searchParams);

// Get the enumeration of result items
IEnumerable<PortalItem> searchResults = searchResultInfo.Results;
  // Iterate thru the results
foreach(PortalItem item in searchResults)
{
    // ... work with items here ...
}
```

`ArcGISPortal` search methods, such as `FindItemsAsync`, `FindGroupsAsync`, and `FindUsersAsync` allow you to provide an optional `CancellationToken` argument that gives the user the ability to cancel execution of the search task.

Copyright © 1995-2020 Esri. All rights reserved.

Note:
Remember, only the items that are accessible to the authenticated user of the portal will be included in the search results. When a user is not authenticated, only publicly shared items are found.

By default, the result set contains only the first 10 results but provides a convenient way to fetch subsequent sets of results if required. To learn more about working with large result sets, see Display many search results.

For a full list of the default fields used for searching, and complete information on constructing portal query strings, see the ArcGIS Online Search reference.

Order the search results

If there are more than a few results, you may want to help your user narrow down the results to find items of most interest. Your app can show the most recently updated results to the user first, sorting search results by the date they were last modified, in descending order.

```
searchParams.SortField = "modified";
searchParams.SortOrder = PortalQuerySortOrder.Descending;
```

Results can be sorted in either ascending or descending order, by portal item fields such as the title, date of creation or date last modified, type, and owner. For a full list of sort fields, see ArcGIS Online Search.

Find items by type

You can restrict the results of a search to be a specific type or types of content; for example, you can search to find only web maps.

The expression used to set the `QueryString` property can further refine search criteria using specific fields as well. In your query expression, use a property identifier (such as `orgid`, `owner`, `title`, or `type`), followed by a colon and the value. The following expression, for example, will search for `web map` items within the portal organization with id `123ABC` that match the term `Cache`.

```
var searchParams = new Esri.ArcGISRuntime.Portal.PortalQueryParameters("\"Cache\"
orgid:\"123ABC\" type:(\"web map\" NOT \"web mapping application\")");
```

See the ArcGIS Rest API search reference for more information, including the fields available for use in your query expression.

If you want to find results of more than one type of portal item, you can do this by defining your query string.

Find groups by keyword

You can find groups by using a keyword in a similar way as it is used when finding portal items. For groups, the default fields searched include title, description, and tags (as for portal items), but also include owner.

The following example searches for portal groups with the specified name (title) that do not require an invitation to join. A maximum of three results are returned, sorted by group creation date (oldest to newest).

```
// Build a query that searches for a specified group name (Title)
var queryString = string.Format("title:\"{0}\" isinvitationonly:{1}", groupName, "false");

// Create a SearchParameters object, set options
```

Copyright © 1995-2020 Esri. All rights reserved.

```
var searchGroupParameters = new PortalQueryParameters(queryString)
{
    SortField = "created",
    SortOrder = PortalQuerySortOrder.Ascending,
    Limit = 5
};

// Execute the search
var groupResultInfo = await portal.FindGroupsAsync(searchGroupParameters);
var groupSearchResults = groupResultInfo.Results;
```

Find items in a known group

After a search for groups, you might want to list all of the items shared with those groups. Alternatively, you might want to list the items in a group returned from the authenticated users list of groups. If a search result set contains at least one group, to find all the items in that group, issue a second search using the group ID as part of the second query.

Continuing from the previous example, the code below finds items for each result of a group search.

```
// Report groups and some of the items they contain
Debug.Print("Items for Selected Groups");
// Iterate thru groups in the results
foreach (PortalGroup group in groupSearchResults)
{
    // Print the group name
    Debug.Print(group.Title);
    // Define a new search for (a max of three) items with this group ID
    var groupSearchItemsParams = new PortalQueryParameters("group:" + group.GroupId);
    groupSearchItemsParams.Limit = 3;
    // Perform the search and get the results
    var itemResultInfo = await portal.FindItemsAsync(groupSearchItemsParams);
    var itemSearchResults = itemResultInfo.Results;

    // Print the name of each item in the results (if any)
    if(itemSearchResults.Count() == 0) { Debug.Print("\t--No Items"); }
    foreach (PortalItem item in itemSearchResults)
    {
        Debug.Print("\t--" + item.Title);
        SearchResultsList.Items.Add(item);
    }
}
// Output for a search of groups with "Logan" in the name ...
//Items for Selected Groups
//Logan 911
//    --Story Map Tour
//    --Finder
//    --Basic Viewer
//Logan City Council / Alliance Collaboration
//    --No Items
//Logan City Open Data
//    --Logan Park Finder
//    --Logan Planning Scheme Dec 2015
//    --Logan City Sharps Disposal Locations
```

Find items by ID

If you already know the ID of a portal item, you can search for it by using that ID. You might find this technique useful if your app is closed while displaying a specific item—you can save the ID and fetch the item again to ensure the item is still accessible and all the properties are up-to-date, and use this information to restore the state of the app. If the search returns no results,

Copyright © 1995-2020 Esri. All rights reserved.

then the item may no longer be available; for example, the owner may have changed the access permissions, and your app can react accordingly.

You can create a query string to find an item using its (unique) id as shown in the following example.

```
var portal = await ArcGISPortal.CreateAsync();
// Create a new SearchParameters with a query string that specifies the item ID
var searchParams = new PortalQueryParameters("id:" + itemId);
// Search the portal for an item matching the id
PortalQueryResultSet<PortalItem> searchResultInfo = await portal.FindItemsAsync(searchParams);

// Get the enumeration of result items
IEnumerable<PortalItem> searchResults = searchResultInfo.Results;
// There should either be one or zero items in the results
PortalItem item = searchResults.FirstOrDefault();
if (item == null) { return; }

// ... work with the item here ...
```

You can also open an `ArcGISPortalItem` directly using the static `CreateAsync` method on the `ArcGISPortalItem` class, as shown in the following example.

```
var portalItem = PortalItem.CreateAsync(portal, itemId);
```

Define a query

If you need to search for items using more complex criteria, you can build a query to satisfy that criteria making use of multiple predicates, Boolean operators, ranges, and wildcards. You could, for example, build a query that finds items that are web maps, owned by the user "esri", with either of the tags "atlantic" or "pacific".

```
var portalSearchParams = new PortalQueryParameters("type:\"web map\" AND owner:esri AND
tags:(pacific OR atlantic)");
```

Adding specific criteria can be especially useful when you are localizing your app, as you can limit results to a specific culture.

```
portalSearchParams = new PortalQueryParameters("culture:fr-FR AND type:\"map service\"");
```

Tip:

You can also search using ranges of dates that items were created and last modified, using the created and modified fields of a portal item. However, be aware that portal searches use UNIX epoch dates (defined as the number of seconds that have elapsed since midnight January 1, 1970), as noted in the ArcGIS Online Search reference.

Search for featured basemaps, content, and groups

Portal administrators can configure a number of settings to highlight specific content to users. Configurable settings include the set of available basemaps, content featured in the portal homepage and gallery, and featured groups for the organization.

Get Home page and Gallery featured content

Administrators can configure two groups that contain featured content to show in different situations. The portal gallery can be populated either with the contents of a group, or with the top 100 most-viewed items. The homepage of the organization can

Copyright © 1995-2020 Esri. All rights reserved.

also display featured content from a chosen group, and the maximum number of items displayed can be separately configured. Depending on the type of content that is featured, you may want your app to display the same featured items as shown in the website. For example, if the featured items are web maps, app users may want to browse a list of these maps, and then open them.

The simplest way to search for the portal items featured as homepage content is to use the `Portal.GetHomePageFeaturedContentAsync` method. Note that this returns all of the featured items and is not limited by the number of items the portal administrator may have configured to be shown. The featured content count can be found with `PortalInfo.HomePageFeaturedContentCount`.

```
// Open the portal
ArcGISPortal portal = await ArcGISPortal.CreateAsync();

// Get the featured content
IEnumerable<PortalItem> featuredContent = await portal.GetHomePageFeaturedContentAsync();

// Use Linq to only take as many items as are specified by the portal for display
featuredContent = featuredContent.Take(portal.PortalInfo.HomePageFeaturedContentCount);
```

Similarly, the `GetFeaturedItemsAsync` method returns the set of featured items shown in the gallery.

```
// Open the portal
ArcGISPortal portal = await ArcGISPortal.CreateAsync();

// Get the featured content
IEnumerable<PortalItem> featuredContent = await portal.GetFeaturedItemsAsync();
```

Alternatively, you can find the group or groups configured to contain featured content or items by using `PortalInfo.FeaturedItemsGroupQuery` or `PortalInfo.HomePageFeaturedContentGroupQuery` and using this value to set the query string for a `PortalQueryParameters`. You can then use the query to find the group or groups, and in turn query for the `PortalItems` contained. One advantage of this approach is that query results are returned in batches, suitable for paging.

```
// Open the portal
ArcGISPortal portal = await ArcGISPortal.CreateAsync();
// Get the portal info
PortalInfo info = portal.PortalInfo;
// Set up the featured content group query
PortalQueryParameters featuredContentGroupQuery = new
PortalQueryParameters(info.HomePageFeaturedContentGroupQuery);
// Get the featured content group
PortalQueryResultSet<PortalGroup> results = await
portal.FindGroupsAsync(featuredContentGroupQuery);


// Take the first group in the results
PortalGroup group = results.Results.FirstOrDefault();
// Return if the result is empty
if (group == null) { return; }

// Set up query for the items in the featured group
PortalQueryParamters itemsSearchParams = new PortalQueryParameters("group:" + group.GroupId);
// Run the search
PortalQueryResultSet<PortalItem> featuredItemsResultInfo = await
```

Copyright © 1995-2020 Esri. All rights reserved.

```
portal.FindItemsAsync(itemsSearchParams);
// Get the result portal items
IEnumerable<PortalItem> featuredItems = featuredItemsResultInfo.Results;
```

Get a list of configured basemaps

Many apps allow a user to change the basemap used to provide context for other data shown in a map. Portal administrators can configure the list of basemaps they want to be available to a user of a portal by sharing web maps into a specific group.

```
// Open the portal
ArcGISPortal portal = await ArcGISPortal.CreateAsync();

// Get the basemaps
IEnumerable<Basemap> basemaps = await portal.GetBasemapsAsync();
```

For each web map in the result set, you can show the thumbnail and title of the item in a gallery or list. You can easily display a web map from the selected item by using its ID.

Additionally, you can use another predefined query (`PortalInfo.BasemapGalleryGroupQuery`) to find out the basemaps group for a portal if required, and from there you can list the items in that group.

Get featured groups

A list of featured groups can be configured to appear on the Groups and My Organization pages of the website. You can find out these groups, and in turn can get the contents as you would for any other group.

```
// Open the portal
ArcGISPortal portal = await ArcGISPortal.CreateAsync();

// Get the groups
IEnumerable<PortalGroup> groups = await portal.GetFeaturedGroupsAsync();
```

Display many search results

Searches may return just a few results, or many thousands of results. To reduce network traffic and memory , search results are returned in batches (by default, only the first 10 are returned), along with a count of the total results found on the portal.

Often, the required item is in the first set of results, but your app can let the user decide if they need to see more items, for example, by scrolling past the end of a list, switching to the next page of results, or by tapping a button. You can easily issue another query to find the next set of results, as the result set from your first query constructs the correct subsequent query for you.

The following code creates a `PortalQueryParameters`, sets the initial query, and calls the `ArcGISPortal.FindItemsAsync` method. After iterating through the initial results, the `PortalQueryResults<T>.NextQueryParameters` method is used to get the correct query to find the next set of results. If this query is not null, then a second call to `FindItemsAsync` is made. This pattern is repeated until `PortalQueryResults<T>.NextQueryParameters` method returns null, indicating there are no more results to get.

```
// See if all results have been processed
if (_gotAllResults) { return; }
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Create a new PortalQueryParameters (if it doesn't exist, this is the first query)
if(_portalSearchParams == null)
{
    // Set the query text and the limit
    _portalSearchParams = new PortalQueryParameters("Cache Valley");
    _portalSearchParams.Limit = 10;
    _portalSearchParams.StartIndex = 1;
}

// Search for portal items using the SearchParameters
PortalQueryResultSet<PortalItem> searchResultInfo = await
_portal.FindItemsAsync(_portalSearchParams);
  // ... work with search results here ...
  // Get the NextQueryParameters so the next search will return the appropriate set of results
_portalSearchParams = searchResultInfo.NextQueryParameters;
  // Set the flag for all results, if NextQueryParameters is null all results have been processed
_gotAllResults = _portalSearchParams == null;
```

Note:
When all results have been returned for a search, `PortalQueryResults<T>.NextQueryParameters` returns **null**.

If you want to retrieve results in larger batches, use `SearchParameters.Limit` to change the batch size.

Related topics

Access the ArcGIS platform
Access portal content
Share a portal item
Add items to a portal

Copyright © 1995-2020 Esri. All rights reserved.

Share a portal item

When one of your users creates a portal item and adds it to a portal, using ArcGIS Online or this ArcGIS Runtime API, the item is private by default. Only the owner of the item (and the organization's administrators) are able to view it. This level of sharing is indicated by the portal item's access property.

```
var isPrivate = (portalItem.Access == PortalAccess.Private);
```

⚠️ **Caution:**

Methods for getting or updating sharing information are only available to the portal item's owner and administrators of the organization. Calling these methods as an unauthorized user will result in an exception.

The access property indicates the item's highest level of sharing. So if the portal item has an access value of:

- Private—The item is not be shared with anybody. Only the owner and administrator can view the item.
- Public—The item can be viewed by everybody.
- Organization—The item can be viewed by every member of the organization that the item is created in.
- Shared—The item can only be viewed by members of a group that the item is shared with.

You can allow users to change this level of sharing so that the portal item is available to the right users.

Share with everyone or a portal's organization

If you want your portal item to be accessible to everyone, you need to make it public. Any user who has access to the portal website can then find and use the portal item. To make a portal item public, call the `ShareWithAsync` method and pass `true` for the `everyone` argument.

```
// Share the portal item with the entire organization, or with everyone
await portalItem.ShareWithAsync(withEveryone, withOrganization);
Console.WriteLine("Sharing for " + portalItem.Title + " has been updated.");
```

To ensure only members of the user's portal have access to an item, you can share it with just the user's portal organization. To do this, pass `true` for the `organization` parameter of the `ShareWithAsync` method.

```
// Share the portal item with the entire organization, or with everyone
await portalItem.ShareWithAsync(withEveryone, withOrganization);
Console.WriteLine("Sharing for " + portalItem.Title + " has been updated.");
```

Share with a portal group

If a user is a member of a group, they can share items that they own with that group. Sharing items with specific groups restricts access to a smaller, more focused set of people. For details on portal groups, see What is a group? The following sections describe how to:

- Find the groups that the portal item belongs to
- Find the groups the user is a member of
- Share a portal item with a group

Copyright © 1995-2020 Esri. All rights reserved.

- Unshared a portal item from a group

Find a portal item's groups

You can determine which groups a portal item is currently shared with by calling the `GetGroupsAsync` method on the portal item. This asynchronous method returns three types of groups. In this example:

- `adminGroups` contains the groups which the current user is an administrator of.
- `memberGroups` contains the groups in which the user is a member.
- `otherGroups` contains the public groups or groups that are shared to the user's organization but that the user is not a member of.

This code displays the title of the groups that the item is shared with.

```
// Get the groups that have access to this item
var itemGroups = await portalItem.GetGroupsAsync();
var adminGroups = itemGroups.Admins;
var memberGroups = itemGroups.Members;
var otherGroups = itemGroups.Others;
// Report the user's role in each group
var sb = new StringBuilder();
sb.AppendLine("Groups for this item");
// Admin
sb.AppendLine("You are an admin in these groups:");
foreach(var group in adminGroups)
{
    sb.AppendLine("- " + group.Title);
}
// Member
sb.AppendLine("You are a member of these groups:");
foreach (var group in memberGroups)
{
    sb.AppendLine("- " + group.Title);
}
// Public group, not a member
sb.AppendLine("You have access to these groups:");
foreach (var group in otherGroups)
{
    sb.AppendLine("- " + group.Title);
}

Debug.Print(sb.ToString());
```

Share a portal item with specific groups

To share a portal item with specific groups, first determine which groups it should be shared with. For example, you can determine which groups the current user is a member of by iterating through the groups returned from the `PortalUser.Groups` property as follows:

```
// Get all groups to which the current user belongs
var usersGroups = myPortal.User.Groups;
// Loop through all the user's groups
foreach(var g in usersGroups)
{
    // Show the title (name) of each group
    Console.WriteLine(g.Title);
}
```

Copyright © 1995-2020 Esri. All rights reserved.

Note that some groups are view-only; that is, members can view the items already in the group, but cannot share other items with that group. To learn more about users and groups, see List the groups a user belongs to and Portals, users, roles, groups, and sharing.

Then to share the portal item with those groups, pass a list of these groups to the `ShareWithGroupsAsync` method as follows:

```
// Share the portal item with specified groups (using an enumeration of GroupInfo)
await portalItem.ShareWithGroupsAsync(groupIdList);
Console.WriteLine(portalItem.Title + " is now shared with the selected groups.");
```

If there is a problem sharing the portal item with some groups, the failures will be indicated in the values returned from the `ShareWithGroupsAsync` method.

Unshare a portal item from specific groups

To stop sharing a portal item with specific groups pass a list of these groups to the `UnshareGroupsAsync` method as follows:

```
// Unshare the portal item with a list of groups
// Query for all groups owned by Ted Bell
PortalQueryParameters groupQueryParams = new PortalQueryParameters("owner:(\"Ted_Bell\")");
PortalQueryResultSet<PortalGroup> portalGroups = await
myPortal.FindGroupsAsync(groupQueryParams);

// Unshare the portal item with all the groups in the results
await portalItem.UnshareGroupsAsync(portalGroups.Results);
Console.WriteLine("No longer sharing " + portalItem.Title + " with Ted's group(s).");
```

Stop sharing

To stop sharing the item, call the `UnshareAsync` method. Calling this method:

- Makes the item private and accessible to only the item owner (and the organization's administrator)
- Removes all groups from the item's groups list

```
// Unshare item from all groups with which the item is currently being shared
await portalItem.UnshareAsync();

// Set sharing to false for the organization and for everyone
await portalItem.ShareWithAsync(false, false);
```

Related topics

Access the ArcGIS platform
Access portal content
Search for content
Add items to a portal

Copyright © 1995-2020 Esri. All rights reserved.

Add items to a portal

Portals, such as ArcGIS Online and ArcGIS Enterprise, can store many types of items. Most types are directly related to the ArcGIS platform, including ArcGIS documents (.mxd, .3dd, .mapx), datasets and packages (.lyr, .kml, .tpk), service urls, and so on. You can also store things like Microsoft Office documents (.docx, .pptx, .xlsx), comma-separated values (.csv), portable document format (.pdf), and a variety of image formats (.jpg, .png, .tif). Perhaps the most familiar type of portal item is a web map — a JSON description of the data in a map, along with other display and behavioral properties. Web maps are particularly useful in your ArcGIS Runtime SDK apps, as they provide a way for you to display a consistent set of data and symbology to your users. Portal items also offer the ability to make and save changes to these items, if needed.

The `PortalItem` class represents an individual item stored in a portal. Each item includes the following:

- Metadata—such as a unique identifier (ID), title, summary, description, thumbnail image, and tags
- Binary or text-based data—for example, the JSON that defines a web map or a feature collection, or a URL pointing to a map service, or the binary data of a tile package or shapefile
- Sharing settings—the access level and groups to whom the item is accessible

This topic will describe creating new portal items in general, setting common properties, and adding them to a portal. For information specific to working with web map portal items, see the Display a map and Save a map topics. For information about setting the right access level for your items, see Share a portal item.

Create a new portal item

A new portal item must be created in the context of a particular portal, such as your ArcGIS Online organization. The user connecting to the portal must also have permissions to create new items. See the Access the ArcGIS Platform topic for information about connecting to a portal as an authenticated user. Once you've connected to your portal, you can create a new portal item by specifying the portal on which it will be hosted, the type of data it will store, and a name (title) to display for the item. The item content can be specified when initially adding the item to the portal, or can be updated later. A new portal item can be added to the root folder, or to any available subfolder for the currently authenticated user.

See the `Esri.ArcGISRuntime.Portal.ItemType` enumeration for a list of available portal item types.

The following example creates a new portal item called "TES Species Summary" that will store a PDF document. It sets properties to provide a summary (short description), a description, and a set of tags to improve searchability. The code then adds the new item to the root folder on the user's portal.

```
// (myPortal is a connection to a portal as a user with permission to create new portal items)
var item = new PortalItem(myPortal, PortalItemType.PDF, "TES Species Summary")
{

    // the Snippet property provides a brief summary for the item
    Snippet = "2015 TES summary report",
    // use the Description to describe the item in more detail
    Description = "The 2015 summary report describing the status of threatened, endangered, and
sensitive species within the Wakim National Forest."
};
// provide relevant tags to help users find the item
item.Tags.Add("TES");
item.Tags.Add("Wakim National Forest");
item.Tags.Add("Endangered species");

// add the new item (without content) to the portal
var newPortalItem = await myPortal.User.AddPortalItemAsync(item);
```

Copyright © 1995-2020 Esri. All rights reserved.

Note:
Don't be confused by the static `PortalItem.CreateAsync` method, which is used to instantiate a portal item in your code using the unique ID for an existing item on the portal. Despite what the name may imply, it is not used to create a brand new portal item.

When adding a new item to a portal, you can specify the item content and a subfolder in which to store the item. The available subfolders are those that belong to the currently connected portal user. The following example gets a list of the available subfolders in which a new portal item can be stored.

```
// fill a list box with the user's folder names
var myStuff = await myPortal.User.GetContentAsync();
foreach (var folder in myStuff.Folders)
{
    PortalFoldersListBox.Items.Add(folder);
}
```

The following example adds a new portal item into a folder selected by the user.

If you add an empty portal item, you can load the content (data) for the item later as described in the following section.

Update a portal item

A portal item can have content that is based on simple text or on binary data. The content for a portal item may be read from a local file as text (for a CSV item, for example), as binary data (for an image file, for example), or simply set with a URL (for GIS services). You can assign content to a portal item when initially adding it to the portal (`PortalUser.AddPortalItemAsync`) or you can update a portal item's content later using `UpdateDataAsync`, as shown in the following example.

```
// read data (FileStream) from the local PDF file
var pdfData = File.OpenRead(reportFilepath);
 // update the portal item with the PDF content, close the PDF file
await newPortalItem.UpdateDataAsync(pdfData);
pdfData.Close();
```

A portal item thumbnail image can also be set as part of the content, as shown in the following example.

```
// create a thumbnail image for the portal item
var thumbnailData = new FileStream(@"D:\Temp\TES-Report.jpg", FileMode.Open);
newPortalItem.SetThumbnail(thumbnailData);

// update the portal item with the thumbnail, close the image file
await newPortalItem.UpdateItemPropertiesAsync();
thumbnailData.Close();
```

Copyright © 1995-2020 Esri. All rights reserved.

For best results, a thumbnail image should be 200 pixels wide by 133 pixels high (other sizes will be adjusted to fit). Acceptable thumbnail image formats are: PNG, GIF and JPEG. The maximum file size is 1Mb

Note:
Portal item metadata, such as its title, summary, description, and so on can also be updated by calling `UpdateItemPropertiesAsync` to save changes to the item. This method is available to the item owner and administrator of the organization only.

Set sharing

When a user creates a new portal item (using ArcGIS Online or an ArcGIS Runtime API) its accessibility is **private** by default, meaning only the owner of the item (and the organization's administrators) are able to view it. If you choose to, you can share your portal item with:

- The creator of the item and portal administrators
- Your entire organization
- Specified portal groups
- Everyone

Copyright © 1995-2020 Esri. All rights reserved.

Note:
To control sharing for an item, you must be able to connect to the portal as an authenticated user with the appropriate permissions (the item owner or a portal administrator).

You may also choose to share your item among combinations of the above, such as your entire organization and some specified groups, for example. See the Share a portal item topic for more information.

Related topics

Access the ArcGIS platform
Access portal content
Search for content
Share a portal item

Copyright © 1995-2020 Esri. All rights reserved.

Use the Authentication Manager

Your app may need to access services that are restricted to authorized users. For example, your organization may host private data layers or feature services that are only accessible by verified users. You may also need to take advantage of premium ArcGIS Online services, such as routing, that require secured user access. The ArcGIS platform provides many ways to secure access to your organization's content and services. The Runtime SDK provides full support for access to secured ArcGIS Server, ArcGIS Online, or ArcGIS for Portal resources using the following authorization methods:

- ArcGIS Tokens: proprietary token-based authentication mechanism
- OAuth 2.0: secure delegated access to server resources
- Network credential: HTTP secured service / Integrated Windows Authentication (IWA)
- Certificate: Public Key Infrastructure (PKI)

Implementing these security methods in your app (potentially for a variety of platforms) can be a lot of work. Fortunately, ArcGIS Runtime SDK provides a component that abstracts much of the authentication logic for you, called `AuthenticationManager`. This class can be configured to react to security issues encountered in your app (such as requests for secured resources) by using a pre-defined credential, prompting the user for the correct type of credential required by the server, or other custom logic you define.

Authentication manager overview

To abstract some of the low-level details for working with secure resources, authentication manager handles some of the details when accessing secured resources in your app, such as challenging the user for credentials when a secured resource is accessed and storing credentials when the user successfully authorizes. It also manages a collection of server information and available user credentials if they have been provided. When a request for a secured resource is made to one of the registered servers, the authentication manager automatically includes the stored credentials in the request or handles getting a credential if one does not exist.

`AuthenticationManager` is a singleton class, which means there is always exactly one instance. You cannot instantiate an object of this type in your code, instead you must reference it using the `Current` static property.

Depending on the authorization method and platform for your app, the authentication manager may require a minimum amount of configuration to handle authenticating a user, including showing its own UI for gathering login information. In some situations, you may need more control over the authentication process by customizing the default handler.

Register servers with the authentication manager

Within your app, authentication manager maintains a collection of information about servers that require authentication to access secured services. You can register a server with the authentication manager by providing its URL and the type of authorization required. Registering a server is not always necessary, as authentication manager can usually determine the URI of a secured server as well as the token endpoint (for ArcGIS Token secured resources, for example) when a secured resource is encountered. For some types of authorization, you must register the server to provide additional information to properly describe authorization parameters for the server. OAuth authentication, for example, requires a unique ID for the client app and a URL to handle the response (and may require a client secret as well). In general, you register these servers when your app initializes to ensure that the authentication manager knows how to handle requests for their services.

The following example registers a server with the authentication manager with an OAuth (implicit) authentication type and defines some required information about the client app.

Copyright © 1995-2020 Esri. All rights reserved.

```
// Register a portal that uses OAuth authentication with the AuthenticationManager
Esri.ArcGISRuntime.Security.ServerInfo serverInfo = new ServerInfo
{
    ServerUri = new Uri("https://www.arcgis.com/sharing/rest"),
    TokenAuthenticationType = TokenAuthenticationType.OAuthImplicit,
    OAuthClientInfo = new OAuthClientInfo { ClientId = "2HEtx9ujil5rac8K", RedirectUri = new
Uri("https://developers.arcgis.com") }
};

AuthenticationManager.Current.RegisterServer(serverInfo);
```

⚠ **Caution:**

If using the `Xamarin.Auth` component in an Android or iOS app, the redirect URL for OAuth authorization must represent a valid endpoint. Arbitrary URLs, such as `http://my.app.redirect` or `urn:ietf:wg:oauth:2.0:oob`, for example, will return a **400 - Bad Request** error. For iOS apps, all OAuth related URLs should be **https**.

If a request for a secured resource is made by your app, authentication manager includes a credential (if available) or makes the appropriate **challenge** for one.

Challenge the user for credentials

A challenge by the authentication manager occurs when a request is made to access a secured resource for which the authentication manager has no credential. A challenge can be resolved (handled, in other words) by providing the appropriate credential either programmatically or by prompting the user. A **challenge handler** associated with the authentication manager contains the logic to determine how to resolve challenges for secured resources.

When a request is made for a secured resource, the authentication manager determines whether the request is to one of its registered servers. If a credential is already available for the server, the authentication manager automatically includes it with the request. Otherwise, the challenge handler logic determines how to resolve the challenge, perhaps by automatically generating a credential for the service or (more commonly) by prompting the user for login information according to the authorization method that's been defined for the registered server. This type of challenge is referred to as an **on-demand** challenge, because it only occurs when a request to a secured resource is made.

The following code will cause a challenge if the server is secured and does not allow anonymous (public) access.

```
// this code may cause an on-demand challenge or result in anonymous access
ArcGISPortal portal = await ArcGISPortal.CreateAsync(new Uri(SERVER_URI));
```

Copyright © 1995-2020 Esri. All rights reserved.

A server, however, often contains both secured and public (unsecured) resources. If the server in the preceding example allows public access for some of its resources, an on-demand challenge is not raised by the authentication manager. Instead, the user would simply be granted anonymous access. To explicitly challenge the user for a specified server, you need to issue a **proactive** challenge.

The following example creates a `CredentialRequestInfo` that defines the server to authenticate with. A proactive challenge is then issued by calling `AuthenticationManager.GetCredentialAsync()`.

```
// define the credential request
CredentialRequestInfo cri = new CredentialRequestInfo
{
    // token authentication
    AuthenticationType = AuthenticationType.Token,
    // define the service URI
    ServiceUri = new Uri(ServerUrl),
    // OAuth (implicit flow) token type
    GenerateTokenOptions = new GenerateTokenOptions
    {
        TokenAuthenticationType = TokenAuthenticationType.OAuthImplicit
    }
};

// issue a proactive challenge by explicitly getting a credential
Credential cred = await AuthenticationManager.Current.GetCredentialAsync(cri, false);
```

The details of how a challenge for a credential is handled (user name and password, client certificate, hard-coded, entered by the user, and so on) are handled by the following components associated with the authentication manager.

- Challenge handler - a component that implements the `IChallengeHandler` interface and is assigned to the `AuthenticationManager.ChallengeHandler` property. The challenge handler contains logic for obtaining the required credential for a server, either by providing hard-coded login information or prompting the user to enter it.
- OAuth authorize handler - a component that implements the `IOAuthAuthorizeHandler` interface and is assigned to the `AuthenticationManager.OAuthAuthorizeHandler` property. The OAuth authorize handler contains logic that's specific to handling authorization using OAuth.

In some situations, you can use the default handlers and utility classes provided by ArcGIS Runtime SDK. If needed, you can also create custom handlers by implementing the required code.

A custom `ChallengeHandler` component can be created by defining a function to challenge for credentials with the signature: `private async Task<Credential> ChallengeFunction(CredentialRequestInfo info)`. The following example shows a function with this signature that gets credentials from the user (by calling another function).

```
private async Task<Credential> Challenge(CredentialRequestInfo info)
{
    // Call code to get user credentials
    // Make sure it runs on the UI thread
    if (this.Dispatcher == null)
    {
        // No current dispatcher, code is already running on the UI thread
        return await GetUserCredentialsFromUI(info);
    }
    else
    {
        // Use the dispatcher to invoke the challenge UI
        return await this.Dispatcher.Invoke(() => GetUserCredentialsFromUI(info));
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    }
}
```

To assign this function as the `AuthenticationManager.ChallengeHandler`, you can create a new `Esri.ArcGISRuntime.Security.ChallengeHandler` helper class (that implements `IChallengeHandler`) and pass in the function as an argument. The following example creates the challenge handler and assigns it to the authentication manager.

```
// Define a method to challenge the user for credentials when a secured resource is encountered
AuthenticationManager.Current.ChallengeHandler = new
Esri.ArcGISRuntime.Security.ChallengeHandler(Challenge);
```

For a complete example of a challenge handler that prompts the user for a username and password, see the Token Challenge sample.

Add or remove credentials

The authentication manager maintains a collection of credentials, and the servers with which they are associated, for use by your app. You can add or remove credentials in this collection as needed. After successfully generating a credential, you must add it to the authentication manager's credential collection. Only credentials in the collection will be used to access secured resources.

The following example generates a new credential using login information entered by the user, then adds the credential to the authentication manager. Notice that the credential has an associated server URI.

```
try
{
    // exception is thrown here if a bad login is entered ...
    var cred = await
Esri.ArcGISRuntime.Security.AuthenticationManager.Current.GenerateCredentialAsync(
        new Uri(PORTAL_SERVER_URL), UserTextBox.Text, PasswordTextBox.Password);

    // add the credential if it was generated successfully
    Esri.ArcGISRuntime.Security.AuthenticationManager.Current.AddCredential(cred);

    // connecting to the portal will use an available credential (based on the server URL)
    _portal = await Esri.ArcGISRuntime.Portal.ArcGISPortal.CreateAsync(new
Uri(PORTAL_SERVER_URL));
}
catch(ArcGISWebException webExp)
{
    msg = "Could not log in. Please check credentials. Error code: " + webExp.Code;
}
```

⚠️ **Caution:**

If the user name and password cannot be authenticated, an `Esri.ArcGISRuntime.Http.ArcGISWebException` is thrown. Make sure you handle such exceptions, as shown in the previous example, especially if you are getting login information from the user. One technique is to ignore such exceptions and connect anonymously (if supported by the server) when authentication fails.

If you no longer need the credential for a server in the authentication manager, you can remove it from the collection. This would be equivalent to signing out of a portal. The credential for a particular server can be found in the collection using the server URI.

Copyright © 1995-2020 Esri. All rights reserved.

The following example finds the credential for a server and removes it from the collection.

```
// find the credential for this server
Credential cred = Esri.ArcGISRuntime.Security.AuthenticationManager.Current.FindCredential(new
Uri(PORTAL_SERVER_URL));

// remove it from the identity manager
Esri.ArcGISRuntime.Security.AuthenticationManager.Current.RemoveCredential(cred);
```

Work with the authentication manager

Authentication manager is a singleton class, which means there is always one for your ArcGIS Runtime app. It can be used to provide authentication for a variety of servers and authorization types because each server you register with authentication manager defines its authorization method. Depending on the authorization type for each server registered, different steps are required to configure authentication manager to challenge for the appropriate credentials. Authentication manager can also have only a single challenge handler associated with it.

ArcGIS Token authorization

For ArcGIS Token authorization, you do not usually need to register the server containing secured resources with authentication manager. If the location (URL) of the server or token endpoint is non-standard, however, you may need to register the server to ensure authentication manager can work with it as expected. In the server info for the registration, indicate the server URI and the token endpoint URI.

The following code registers a server that uses ArcGIS Token authorization.

```
private void RegisterMyTokenServer()
{
    // create a ServerInfo to describe the server and its authorization info
    ServerInfo tokenServerInfo = new ServerInfo
    {
        ServerUri = new Uri("http://my.server.com/"),
        TokenAuthenticationType = TokenAuthenticationType.ArcGISToken,
        TokenServiceUri = new Uri("https://my.server.com/get/some/tokens")
    };
    // register the server with AuthenticationManager
    AuthenticationManager.Current.RegisterServer(tokenServerInfo);
}
```

Challenge handler

To respond to a request for an ArcGIS Token secured resource, you need to have a challenge handler associated with authentication manager. If challenges are consistent — for example, they are always coming from ArcGIS token secured servers and always require a response that prompts the user for a login — your challenge handler can execute the same logic for each secure request. You may, however, have a variety of authorization types for secured resources and different ways of resolving each challenge. In that case, you may need a branching statement in your challenge handler to determine how credentials will be provided.

The following example shows a challenge handler that uses a hard-coded credential based on a known user (username and password). It checks the service URI to see if it's the expected resource and then provides credentials automatically without prompting the user. Additional logic is needed if challenges for other services are to be handled, such as prompting the user for a login. This method would be defined as the `AuthenticationManager` challenge handler using code like this:

Copyright © 1995-2020 Esri. All rights reserved.

```
AuthenticationManager.Current.ChallengeHandler = new
Esri.ArcGISRuntime.Security.ChallengeHandler(CreateKnownCredentials);
```

```
// Challenge method that checks for service access with known (hard coded) credentials
private async Task<Credential> CreateKnownCredentials(CredentialRequestInfo info)
{
    // If this isn't the expected resource, the credential will stay null
    Credential knownCredential = null;

    try
    {
        // Check the URL of the requested resource
        if (info.ServiceUri.AbsoluteUri.ToLower().Contains("usa_secure_user1"))
        {
            // Username and password are hard-coded for this resource
            // (Would be better to read them from a secure source)
            string username = "user1";
            string password = "user1";

            // Create a credential for this resource
            knownCredential = await AuthenticationManager.Current.GenerateCredentialAsync
                                    (info.ServiceUri,
                                        username,
                                        password,
                                        info.GenerateTokenOptions);
        }
        else
        {
            // ... perhaps prompt the user here if the username and password are not known ...
        }
    }
    catch (Exception ex)
    {
        // Report error accessing a secured resource
        _errorMsg = "Access to " + info.ServiceUri.AbsoluteUri + " denied. " + ex.Message;
    }

    // Return the credential
    return knownCredential;
}
```

⚠️ **Caution:**

The preceding example provides a simple illustration of using hard-coded values for a known user. Including usernames and passwords in your code is obviously not a good practice. A better approach is to read them from a secure source, such as a secured web service, encrypted file, and so on.

OAuth authorization

Because OAuth authorization requires additional information about the server, you must register servers using OAuth authorization with the authentication manager. The following types of OAuth authorization methods (grants) are available:

- Implicit: registration requires a client ID and a redirect URL. The user enters a login with the server and receives an access token.
- Authorization code: registration requires a client ID, client secret, and a redirect URL. The user enters a login with the server and receives an access token and refresh token.

Copyright © 1995-2020 Esri. All rights reserved.

- Client credentials: registration requires a client ID, client secret, and a redirect URL. The user does not need to enter a login, but rather is granted access on behalf of the app.

A description of the OAuth grant types is available at oauthlib.readthedocs.io.

The following example registers a server for OAuth with the authorization code grant. The variables `ServerUrl`, `ClientId`, `ClientSecret`, and `RedirectUrl` are string constants that have been defined earlier in the code.

```
// define server info for an OAuth (authorization code) server
var serverInfo = new ServerInfo
{
    ServerUri = new Uri(ServerUrl),
    TokenAuthenticationType = TokenAuthenticationType.OAuthAuthorizationCode,
    OAuthClientInfo = new OAuthClientInfo
    {
        ClientId = ClientId,
        ClientSecret = ClientSecret,
        RedirectUri = new Uri(RedirectUrl)
    }
};

// register the portal with authentication manager
AuthenticationManager.Current.RegisterServer(serverInfo);
```

Caution:

If using the `Xamarin.Auth` component in an Android or iOS app, the redirect URL for OAuth authorization must represent a valid endpoint. Arbitrary URLs, such as `http://my.app.redirect` or `urn:ietf:wg:oauth:2.0:oob`, for example, will return a **400 - Bad Request** error. For iOS apps, all OAuth related URLs should be **https**.

Challenge handler

For OAuth authorization, authentication manager must have an associated challenge handler to respond to requests for secured resources on the server. The logic used in the challenge handler might differ in accordance with the OAuth grant type used (especially for a client credentials grant, in which the user does not need to provide a username and password). Usually, a challenge handler for OAuth will simply ask the authentication manager to generate a credential for the server, which will launch a login UI for the server (as described below).

The following example is a challenge handler method for a server using OAuth (either implicit or authorization code grant type). This method would be defined as the `AuthenticationManager` challenge handler using code like this:

`AuthenticationManager.Current.ChallengeHandler = new Esri.ArcGISRuntime.Security.ChallengeHandler(CreateCredentialAsync);`

```
// ChallengeHandler function for AuthenticationManager, called when access to a secure resource
is attempted
public async Task<Credential> CreateCredentialAsync(CredentialRequestInfo info)
{
    OAuthTokenCredential credential = null;

    try
    {
        // Create generate token options if necessary
        if (info.GenerateTokenOptions == null)
        {
            info.GenerateTokenOptions = new GenerateTokenOptions();
        }
```

Copyright © 1995-2020 Esri. All rights reserved.

```
        // Call GenerateCredentialAsync, pass in the server URI and token options
        // (AuthenticationManager will handle challenging the user for credentials)
        credential = await AuthenticationManager.Current.GenerateCredentialAsync
        (
            info.ServiceUri,
            info.GenerateTokenOptions
        ) as OAuthTokenCredential;
    }
    catch (Exception ex)
    {
        // Exception will be reported in calling function
        throw (ex);
    }

    // Return the credential (or null if authentication failed)
    return credential;
}
```

The call to `AuthenticationManager.Current.GenerateCredentialAsync()` in the previous code will cause authentication manager to show a UI to log in to the specified server, as described in the following section.

OAuth authorization handler

A request to generate credentials for a resource secured with OAuth causes the authentication manager to look for an OAuth authorization handler to show a login UI for the specified server. For some platforms (such as Universal Windows Platform), ArcGIS Runtime has a built-in component that serves as the OAuth authorization handler. On others (such as Windows Presentation Framework) you must create a custom component to present the login UI to the user and get an access token from the server.

An OAuth authorization handler is a class that implements the `Esri.ArcGISRuntime.Security.IOAuthAuthorizeHandler` interface. This interface defines a single method with the following signature.

```
// Function to handle authorization requests, takes URIs for the secured service, the
authorization endpoint, and the redirect URI
public Task<IDictionary<string, string>> AuthorizeAsync(Uri serviceUri, Uri authorizeUri, Uri
callbackUri)
```

The OAuth authorization handler shows the login UI (generated by the server that hosts secure content) in a web browser or control. When the user logs in successfully, the component is responsible for obtaining the authorization from the server. If the user cancels the login or closes the window without continuing, it should raise the appropriate exception.

To assign your custom `IOAuthAuthorizeHandler` class as the OAuth authorization handler, set a new instance of the class to the `AuthenticationManager.Current.OAuthAuthorizeHandler` property. For an example of a custom `IOAuthAuthorizeHandler` component to handle OAuth logins for a WPF app, see the OAuth security sample on GitHub.

For the Android, iOS, and Xamarin Forms platforms, the `Xamarin.Auth` assembly (available via a NuGet package) contains components to show an OAuth login UI. These components can be used with ArcGIS Runtime to create an OAuth authorization handler. See the iOS OAuth security sample on GitHub for an example.

Copyright © 1995-2020 Esri. All rights reserved.

HTTP (integrated Windows authentication) authorization

Using HTTP basic, HTTP digest or Integrated Windows Authentication (IWA) authorization, resources are protected by user name and password set on the service and prompted by browser popup or session cookie. When you use IWA, logins are managed through Microsoft Windows Active Directory. Users do not sign in and out of the portal website; instead, when they access resources, they are signed in using the same accounts they use to log in to Windows. A portal secured with HTTP authorization does not need to be registered with the authentication manager. A response from the server for a request for a secured resource will indicate the authorization mode implemented by the server.

When accessing an item secured with IWA on some platforms (such as Windows Presentation Framework), default credentials (the login that started the app, in other words) are automatically included and the user is never prompted to login. On other platforms such as Android, iOS, and Universal Windows Platform (UWP) credentials must be entered explicitly, either by prompting the user for them or by hard-coding them into the app.

The following example adds an `ArcGISNetworkCredential` with username, password, and domain values entered by the user to `AuthenticationManager`. `NetworkUsername`, `NetworkPassword`, and `NetworkDomain` are public variables that were defined earlier in the code (perhaps from a UI presented to the user on start up).

```
// Create a hard-coded network credential
ArcGISNetworkCredential hardcodedCredential = new ArcGISNetworkCredential
{
    Credentials = new System.Net.NetworkCredential(NetworkUsername, NetworkPassword,
NetworkDomain),
    ServiceUri = new Uri(SecuredPortalUrl)
};

// Add the credential to the AuthenticationManager and report that a non-default credential is
being used
AuthenticationManager.Current.AddCredential(hardcodedCredential);
```

Challenge handler

For platforms that provide a built-in mechanism for showing a UI for entering a login, a custom challenge handler is not required by the authentication manager. Likewise, for platforms that include the current Windows credential with requests for secured resources, you might be comfortable with assuming that authorized users will have access and no custom logic to show a login will be required. Otherwise, you will need a challenge handler that shows a login UI (controls for username, password, and domain), creates a credential, and adds it to the authentication manager. If the login is cancelled or unsuccessful, a null credential should be returned.

For an example of using a custom challenge handler and login UI to implement IWA security in your app, see the Xamarin Forms IWA sample on GitHub.

Public Key Infrastructure (PKI) authorization

Public Key Infrastructure (PKI) uses a mathematical technique called public key cryptography to generate the digital keys that represent a user or organization. ArcGIS for Server leverages the PKI solution with the IIS web server using the ArcGIS IIS Web Adaptor. When a request is made for a resource on ArcGIS for Server, the web server authenticates the user by validating a client certificate that contains the user's digital keys. The request (along with the user name) is then forwarded to ArcGIS for Server via the Web Adaptor. ArcGIS for Server verifies that the specified user has access to the requested resource before sending back the appropriate response. A portal secured with PKI authorization does not need to be registered with the authentication manager. A response from the server for a request for a secured resource will indicate the authorization mode implemented by the server.

Copyright © 1995-2020 Esri. All rights reserved.

Challenge handler

For platforms that provide a built-in mechanism for showing a UI for choosing a client certificate (such as Universal Windows Platform), a custom challenge handler is not required by the authentication manager. Otherwise, you need a challenge handler that shows a client certificate UI (a picker that shows available certificates) and creates a credential from it for use in the authentication manager. If the certificate choice is cancelled or unsuccessful, a null credential should be returned.

Copyright © 1995-2020 Esri. All rights reserved.

Design considerations

Copyright © 1995-2020 Esri. All rights reserved.

Performance considerations

The Runtime SDKs are built to be fast, but there are still ways you can improve performance. Performance can be improved through design decisions, by selecting the right hardware and in choices made within your application development environment.

Note:
A video presentation with much of this topic's information is available, Maximizing the Performance of your Apps.

Design decisions

To get the most performance from your app, consider these design decisions when working with feature layers and graphic overlays:

- Number of features, graphics and layers
- Layer ordering
- Geometry symbolization in each layer
- Renderers
- Rendering modes
- Texture compression
- Scale range
- Spatial reference

Numbers of features, graphics and layers

Your app's performance is affected by the number of features, graphics, and layers that can be rendered quickly in a map or scene. The major factors to consider are:

- The complexity of features or graphics
- The specifications of the target device
- Rendering mode (dynamic or static)
- The end-user experience

Typically, maps, and scenes will have in the order of 10s of layers.

Copyright © 1995-2020 Esri. All rights reserved.

Layer ordering

Layers are arranged bottom to top in the map or scene:

- At the bottom is the basemap
- Next are the operational layers
- And on top are any graphics overlays

Within the operational layers grouping and graphics overlays grouping, keeping layers with the same type and rendering mode next to each other, in the layer stack, can improve performance by optimizing data transfer to the GPU.

Layers in static rendering mode are rendered to an image on the CPU. The Runtime will combine static rendering mode layers which are adjacent (in the layer stack) and push the stack as a single set of textures to the GPU. In contrast, when static layers are not adjacent to one another (in the layer stack), if interleaved with dynamic layers, the Runtime must generate multiple images to preserve visibility ordering. In this case, multiple sets of textures must be pushed for each extent change, thereby negatively impacting performance.

Layer order does not affect layers in dynamic rendering mode, because all resources for the layers reside on the GPU, and the GPU can easily arrange these items when drawing.

Geometry symbolization in each layer

While feature layers are limited to a single geometry type, graphics overlays can have any number of geometry types. If you are rendering many thousands of mixed geometry types (points, lines and polygons) in the same graphics overlay, you may get improved application performance by separating the graphics into graphic overlays which contain one geometry type.

Similarly, because the Runtime will try to reuse duplicate symbols, you should limit the number of different types of symbols for features or graphics within a single layer. Performance gains will be particularly noticeable in layers which contain large numbers of features or graphics.

For larger numbers of features or graphics, you should always use a renderer, which guarantees the reuse of symbol instances.

Renderers

You can assign both symbols and renderers to feature layers and graphics overlays. You can still assign unique symbols to each graphic (in which case the symbol takes precedence) without a notable drop in performance, if the layers contain only a small number of features or graphics. The Runtime will reuse duplicate symbols where it can, but there is an overhead cost in this calculation.

You can also add features or graphics more efficiently using a renderer.

Note:
Some renderers (heatmap renderer, for example) don't support dynamic rendering mode and will silently revert to static rendering mode, if set to dynamic rendering mode.

Rendering modes

You can render both feature layers and graphics overlays in either dynamic or static rendering mode. By default:

- Graphics overlays render in dynamic rendering mode
- Point feature layers render in dynamic rendering mode

Copyright © 1995-2020 Esri. All rights reserved.

- Polyline and polygon feature layers render in static rendering mode

Should you choose to set the rendering mode yourself, keep the following considerations in mind:

- Dynamic rendering mode: In this mode, features and graphics are stored on the GPU. As a result, dynamic rendering mode is good for moving objects and for maintaining graphical fidelity during extent changes, since individual graphic changes can be efficiently applied directly to the GPU state. This gives the map or scene a seamless look and feel when interacting with it. The number of features and graphics has a direct impact on GPU resources, so large numbers of features or graphics can affect the responsiveness of maps or scenes to user interaction. Ultimately, the number and complexity of features and graphics that can be rendered in dynamic rendering mode is dependent on the power and memory of the GPU.
- Static rendering mode: This mode renders features and graphics when needed (for example, after an extent change) and offloads a significant portion of the graphical processing onto the CPU. As a result, less work is required by the GPU to draw the graphics, and the GPU can spend its resources on keeping the UI interactive. Use this mode for stationary graphics, complex geometries, and very large numbers of features or graphics. The number of features and graphics has little impact on frame render time, meaning it scales well, and pushes a constant GPU payload. However, rendering updates is CPU and system memory intensive, which can have an impact on device battery life.

Should you choose to specify the rendering mode, the rendering mode should be set before adding the feature layer or graphics overlay to a map or scene. This is because changing the rendering mode of a layer, after its been added to the a map or scene, forces a complete redraw of the map or scene.

For graphics overlays, setting the rendering mode is as simple as choosing the rendering mode at object instantiation. For feature layers, the rendering mode can be set either on the feature layer itself or on the map or scene's load settings, using the preferred feature rendering mode. Once a feature layer or graphics overlay has been added to a map or scene, the rendering mode should not be switched unless absolutely necessary.

Whether in a feature layer or graphics overlay, it's typically better to use static rendering mode on complex geometries (for example, polygons with a large number of vertices).

Note:
Should you choose to render a complex polygon in dynamic rendering mode, the polygon will be generalized based on the map or scene extent.

Texture compression

Texture compression reduces the file size for image files used for texture. A typical compression technique and file format for textures is JPG, however for mobile platforms especially, use the ETC2 compression technique to improve download times, consume less memory, and improve overall performance.

Scale range

All geographical data has a scale within which the data is both useful and accurate. Limiting the scale (called scale range) at which features or graphics are drawn can also improve performance. For example, you may want to set a scale range to a feature layer showing bus routes, which will be removed from the map or scene when your user zooms out to a country boundary level.

Setting a scale range:

- Makes the map or scene less cluttered and easier to use
- Conserves GPU (in dynamic rendering mode) and CPU (in static rendering mode) resources

Copyright © 1995-2020 Esri. All rights reserved.

Spatial reference

Reprojecting has a computational cost. By choosing the spatial reference early in the design of your app and ensuring that all data and layers are stored in the same spatial reference as your basemap, you can substantially improve performance.

Application development environment

The hardware that you choose may mean that you have more than one application development environment choice. Each of these will provide you with different ways to improve performance.

Asynchronous programming

On mobile devices, where resources are limited and network speed is variable, it is advisable to not perform long running tasks on the main thread.

While multi-threaded architecture allows for handling a number of asynchronous tasks, since threads consume device resources, you must be mindful of the amount of threads that are created.

Hardware

Hardware is designed with functionality and specific needs in mind. Be sure to understand the capabilities of your hardware before adding the complexities of an application environment and the ArcGIS Runtime to your requirements.

Device limitations

Mobile devices have less processor and memory resources compared to standard desktop machines. For this reason, applications need to be developed to run in an efficient manner.

Network speed

Mobile devices often use 3G or sometimes lower speed radio communication networks to obtain and transfer data. The speed of these networks vary but are much slower (in terms of data per second) than wired or wireless networks. Due to this network latency, even small requests can take time to return, which makes your application seem sluggish. Therefore, you need to carefully manage the total amount of data and the number of network requests submitted by your mobile application. As an example, it may be more efficient to send a single large request for data rather than multiple small requests. Changing the layer types, application functions, or the flow of your application can also affect network speed.

If your mobile application users are always in the range of a wireless network, your application can retrieve and submit larger amounts of data. However, even in this scenario, it's always good practice to remember the amount of data and number of requests your application uses, whatever the bandwidth.

By understanding the characteristics of the different types of map layers in the API, you can determine the best layers for your needs and ensure that your application performs for your users.

Network connectivity

Some application users may only have intermittent network access, such as intermittent 4G access due to working in remote areas or daily access to a wireless network for synchronizing data. If this is the case, local storage usage is important. The application can be designed to connect to the server to retrieve data the user needs, then store this data on the local device. Applications need to be developed robustly with this in mind, because the network connection can be dropped anytime. Functions need to fail gracefully, and any long running application transactions may need to be rolled back.

Copyright © 1995-2020 Esri. All rights reserved.

Release your app

Copyright © 1995-2020 Esri. All rights reserved.

License your app

Before you deploy your ArcGIS Runtime app into production, you're required to license it with one of the four ArcGIS Runtime license levels: Lite, Basic, Standard, or Advanced.

Note that other costs may be incurred when deploying your app, such as credits used by ArcGIS Online services in your app. If your app generates revenue, you need to purchase a paid ArcGIS Developer subscription. Refer to our FAQ for more information on the terms and conditions.

Note:
No license is required to develop ArcGIS Runtime apps; there's no need to authorize your development machine. For details on setting up your development environment, see the topic Install the SDK.

Develop and test

You can download and install an ArcGIS Runtime SDK using an ArcGIS for Developer account. You then have access to all of the functionality in the API for development and testing purposes. In this mode, the app will adopt the following behavior:

- Any map or scene will be watermarked with the words `Licensed For Developer Use Only` (as pictured here).

- A message will be written to the console or log that states that the app is `Licensed For Developer Use Only`.

Licensing your app for deployment will remove the watermark and console output and provide the appropriate capabilities to your app users.

Licensing options

Before you can use your app in production, you are required to license the ArcGIS Runtime app. Esri provides four ArcGIS Runtime license levels: Lite, Basic, Standard, and Advanced. Each additional level opens more capabilities for your app users.

Two options are available to set the license level in your app:

Named user

A named user refers to an ArcGIS organizational account that is a member of an organization in ArcGIS Online or ArcGIS Enterprise. The ability to license an ArcGIS Runtime app is just one feature of a named user. You need to include code in your ArcGIS Runtime app to enable a user to sign in (log in) to a portal and return license details associated with their named user account.

Use named users if your target audience has access to ArcGIS Online or ArcGIS Enterprise and use devices that will be online at least once every 30 days (timeout for a named user license for ArcGIS Runtime). One notable benefit of this model is the license for ArcGIS Runtime travels with the named user, not the app, so one named user can license many ArcGIS Runtime apps.

Copyright © 1995-2020 Esri. All rights reserved.

License key

A license key is a text string that contains license details. License keys can be acquired through your ArcGIS for Developer account or purchased in ArcGIS Runtime deployment packs. License keys are usually compiled into your ArcGIS Runtime app. Each installation of your app available to a single user counts as a deployment.

Use a license key if your target audience does not have access to named user accounts with an ArcGIS Online organization or on-premises ArcGIS Enterprise portal, needs to use an application that will remain offline indefinitely, or needs to guarantee an application will function if offline more than 30 days. Customers are required to track the number of ArcGIS Runtime deployments at the Basic, Standard, and Advanced level. Each installation of your app available to a single user counts as a deployment.

Licensing capabilities

The following table shows the functionality associated with each license level, the available licensing options, and how to acquire a license. For details on API classes and members that require a paid license or extension, see Licensing details by API type.

License level	Capabilities	License key	Named user
Lite	Viewing maps, scenes, layers, and packages from the ArcGIS platform. Edit features in public feature services. These services are available on the Internet and are not secured. Generate a mobile geodatabase with features from a sync-enabled feature service. Download updates from a sync-enabled feature service to a mobile geodatabase. Place finding—see the note below regarding storage and the World Geocoding Service from ArcGIS Online. Optimized routing. Calculate service areas and find closest facilities using a network service. Viewing KML data accessed as a web resource (for example, via an http or https link).	Available for free. Log in to your ArcGIS for Developers account, go to your dashboard, and copy the ArcGIS Runtime Lite license key into your app.	With ArcGIS Runtime 100.6 or earlier - • Log in to ArcGIS Online or an on-premises Portal as a Named User of type Viewer, or a level 1 Named User. With ArcGIS Runtime 100.7 or later - • Log in to ArcGIS Online or an on-premises Portal from ArcGIS Enterprise 10.8 or higher as a Named User who has an ArcGIS Runtime **Lite** license. • Log in to an on-premises Portal from ArcGIS Enterprise 10.7 or earlier as a Named User of type Viewer, or a level 1 Named User

Copyright © 1995-2020 Esri. All rights reserved.

License level	Capabilities	License key	Named user
Basic	All capabilities of Lite. Edit features in mobile geodatabases and features in feature services that are only accessible on a local network or are secured. Synchronize edits to feature geometry, attributes, and attachments from a mobile geodatabase to a feature service. Add, update, or delete content on portals. Navigate a route using the route tracker.	Deployment packs available for purchase. Contact the Esri office that serves you or if you are in the United States call Esri at 1-800-447-9778.	With ArcGIS Runtime 100.6 or earlier - • Log in to ArcGIS Online or an on-premises Portal as a Named User of type Editor, Field Worker, Creator, or GIS Professional, or a level 2 Named User. With ArcGIS Runtime 100.7 or later - • Log in to ArcGIS Online or an on-premises Portal from ArcGIS Enterprise 10.8 or higher as a Named User who has an ArcGIS Runtime **Basic** license. • Log in to an on-premises Portal from ArcGIS Enterprise 10.7 or earlier as a Named User of type Editor, Field Worker, Creator, or GIS Professional, or a level 2 Named User.
Standard	All capabilities of Basic. Access to additional data, including shapefiles, GeoPackages, ENC layers (S-57), local raster layers, use of raster functions, and local raster elevation sources. View, create, edit, and save KML data stored as a local file. Visual analysis, including line of site and viewshed.	Deployment packs available for purchase. Contact the Esri office that serves you or if you are in the United States call Esri at 1-800-447-9778.	Log in to ArcGIS Online or an on-premises Portal from ArcGIS Enterprise 10.8 or higher as a Named User who has an ArcGIS Runtime **Standard** license. Note - This option was not available prior to ArcGIS Runtime 100.7
Advanced	Adds Local Server support with Java, .NET and Qt SDKs only.	Deployment packs available for purchase. Contact the Esri office that serves you or if you are in the United States call Esri at 1-800-447-9778.	Log in to ArcGIS Online or an on-premises Portal from ArcGIS Enterprise 10.8 or higher as a Named User who has an ArcGIS Runtime **Advanced** license. Note - This option was not available prior to ArcGIS Runtime 100.7

Extension licenses

ArcGIS Runtime offers extensions that provide access to additional capabilities, analysis tools, and/or data.

Copyright © 1995-2020 Esri. All rights reserved.

Extension license	Capabilities	License key	Named user
Analysis	Minimum ArcGIS Runtime license level: Standard Calculate service areas and find closest facilities using a local network dataset	Deployment packs available for purchase. Contact the Esri office that serves you or if you are in the United States call Esri at 1-800-447-9778.	Log in to ArcGIS Online or an on-premises Portal from ArcGIS Enterprise 10.8 or higher as a named user who has been assigned an ArcGIS Runtime **Analysis** add-on license extension Note - This option was not available prior to ArcGIS Runtime 100.7
Utility Network	Works with all license levels Analyze (trace) utility networks	Deployment packs available for purchase. Contact the Esri office that serves you or if you are in the United States call Esri at 1-800-447-9778.	Log in to ArcGIS Online or an on-premises Portal from ArcGIS Enterprise 10.8 or higher as a named user who has been assigned a **Utility Network** user type extension. Note - This option was not available prior to ArcGIS Runtime 100.7

Copyright © 1995-2020 Esri. All rights reserved.

Extension license	Capabilities	License key	Named user
StreetMap Premium for ArcGIS Runtime	Works with all license levels • Use offline with mobile map packages • Maps with high-quality cartography • Offline Geocoding • Offline Routing StreetMap Premium can be downloaded for these regions: • North America • Latin America • Europe • Asia Pacific • Middle East and Africa • Japan Each regional mobile map package is licensed with an extension license key. You can use any combination of these packages with their associated license keys. **Note:** StreetMap Premium for ArcGIS Runtime licenses are purchased and provided by the customer. You must provide a mechanism for your app to access this license key. The StreetMap Premium data cannot be used after the license has expired. Upon expiration the app user must purchase another license to continue using the data. You must provide the correct messages in your app to notify the user that their license has expired or is due to expire.	Licenses available for purchase by the user of your app. StreetMap Premium for ArcGIS Runtime extension licenses are provided on an annual-term basis. The StreetMap Premium dataset cannot be used after the license has expired. Contact the Esri office that serves you or if you are in the United States call Esri at 1-800-447-9778.	Not available yet.

Store geocode results from ArcGIS Online

The World Geocoding Service from ArcGIS Online includes a `for_storage` parameter. Storing the results of a geocode or reverse geocode operation using this service (by setting the `for_storage` parameter to **true**) requires an ArcGIS Online paid subscription and uses credits from the subscriber's account. Additionally, the user performing a paid operation must be assigned a user role that includes the Geocoding privilege. For more information, see Free vs paid in the REST documentation for this geocode service and User types, roles, and privileges in ArcGIS Online documentation.

Named user licensing

Beginning with the release of ArcGIS Enterprise 10.5, members in an organization in ArcGIS Online or on-premises ArcGIS Enterprise portal can be assigned one of two membership levels, as described in Levels, roles, and privileges. Level 1 can be used to license ArcGIS Runtime at the Lite level. Level 2 can be used to license ArcGIS Runtime at the Lite and Basic level. For Portals prior to version 10.5, all named users are equivalent to Level 2 and can be used to license ArcGIS Runtime at the Basic level.

At the end of 2018, named user types were introduced in ArcGIS Online to replace named user levels. Named users of any type can still be used to license ArcGIS Runtime. Named users of type Viewer can license ArcGIS Runtime at the Lite level and named users of type Editor, Field Worker, Creator, and GIS Professional can license ArcGIS Runtime at the Basic level.

Copyright © 1995-2020 Esri. All rights reserved.

Starting with ArcGIS Enterprise 10.8 and the December 2019 update of ArcGIS Online, named user types explicitly include ArcGIS Runtime licenses and extensions when applicable. ArcGIS Runtime 100.7 and higher is licensed on the basis of these explicit license entitlements when used with such named user types.

Use tile packages from ArcGIS Online basemaps

App developers may use ArcGIS Online basemaps for their apps. Esri has designed a use policy that seeks to maximize the use of these basemaps by the ArcGIS Runtime developer community. The rules for using these basemaps are conditioned on how the basemaps are used.

A developer can only export basemaps from services that support the **export tiles** operation.

A developer may:

- Export basemaps as tile packages for use with an ArcGIS Runtime app, up to the limits imposed by the basemap for a single tile package.
- Use an ArcGIS Runtime app to export basemaps on demand. Authorization must be as a named user or with an app login.
- Embed or bundle basemap tile packages in an ArcGIS Runtime app.
- Use an ArcGIS Runtime app to download pre-created basemap tile packages hosted by ArcGIS Online or ArcGIS Enterprise, but only if the ArcGIS Online organization or ArcGIS Enterprise is licensed for production use.

A developer may not:

- Allow their users to use the exported content outside of ArcGIS technology.
- Allow their users to unbundle the exported content and publish the exported content as a map service.
- Allow their users to unbundle the exported content or redistribute to other users.
- Systematically harvest map tiles using the service (a practice often referred to as "scraping tiles").

License your app

Follow these steps to license your app by with a license key or a Named User account. After this is done, the watermark and debug messages will no longer display and your app will be ready for distribution and use in production.

Note:

If you fail to license the app then the map and scene views will be watermarked with the words `Licensed For Developer Use Only`.

License your app with a license key

Using a license key involves adding code to set the license level at compile time so that the license key is built into the application.

1. Read the end user license agreement.
2. Find a location in your code that runs before any ArcGIS Runtime functionality is used.
3. Call the `SetLicense` method on the `ArcGISRuntimeEnvironment` singleton object to license the app with a License key. The following must run before any other ArcGIS Runtime code.

```
string licenseKey = "runtimelite,1000,rud########,day-month-year,####################";
ArcGISRuntimeEnvironment.SetLicense(licenseKey);
```

Copyright © 1995-2020 Esri. All rights reserved.

Your app is now licensed for deployment.

License your app with a named user account

Use of a named user involves adding code to enable sign-in to using an ArcGIS organizational account. At runtime, a user must sign in successfully to return their license information which will be used by the app to set the license level. Follow these steps to license your app using the named user's account:

1. Read the end user license agreement.
2. Find a location in your code that runs before any ArcGIS Runtime functionality is used.
3. Allow the app user to authenticate with an ArcGIS organizational account. Upon loading the `Portal` obtain the `LicenseInfo` and use this to license the app. As part of the process, save the license information in preparation for your app being used in an offline environment for up to 30 days.
   The following example shows how to get a license from an ArcGIS Online named user. The `AuthenticationManager` has been set up earlier in the code by registering the ArcGIS Online portal, and setting a challenge handler and OAuth authorization handler (as described in the Use the Authentication Manager topic).

```
// Challenge the user for portal credentials (OAuth credential request for arcgis.com)
CredentialRequestInfo loginInfo = new CredentialRequestInfo();

// Use the OAuth implicit grant flow
loginInfo.GenerateTokenOptions = new GenerateTokenOptions
{
    TokenAuthenticationType = TokenAuthenticationType.OAuthImplicit
};

// Indicate the url (portal) to authenticate with (ArcGIS Online)
loginInfo.ServiceUri = new Uri("http://www.arcgis.com/sharing/rest");

try
{
    // Call GetCredentialAsync on the AuthenticationManager to invoke the challenge handler
    Credential cred = await AuthenticationManager.Current.GetCredentialAsync(loginInfo,
false);

    // Connect to the portal (ArcGIS Online) using the credential
    ArcGISPortal arcgisPortal = await ArcGISPortal.CreateAsync(loginInfo.ServiceUri, cred);

    // Get LicenseInfo from the portal
    LicenseInfo licenseInfo = await arcgisPortal.GetLicenseInfoAsync();
    // ... code here to license the app immediately and/or save the license (JSON string) to
take the app offline ...
    // License the app using the license info
    ArcGISRuntimeEnvironment.SetLicense(licenseInfo);
}
catch (Exception ex)
{
    // TODO: handle exceptions
}
```

Note:
The code snippet shown above uses the new `fetchLicenseInfo()` API introduced in ArcGIS Runtime 100.7. For previous releases, use `Portal.portalInfo.licenseInfo` instead. Refer to the documentation from previous releases for full details.

Copyright © 1995-2020 Esri. All rights reserved.

4. If you saved the license information on local storage, your app can be started and licensed in an offline environment using the saved license information. Retrieve the license from storage and license your app.
   After getting the `LicenseInfo` from the portal, you can convert it to a JSON-formatted string. The license string can be stored locally and used to re-create the license as needed, as shown in the following example.

```
// Get the LicenseInfo as JSON formatted text
string licenseJson = licenseInfo.ToJson();
// ... save text string locally ...

// (on startup) ... get the license JSON stored on the device ...
// Set the license info from the saved JSON string
LicenseInfo license = LicenseInfo.FromJson(licenseJson);
```

Extension level licenses

1. Read the end user license agreement.
2. Find a location in your code that runs before any ArcGIS Runtime functionality is used.
3. Call the `SetLicense` method on the `ArcGISRuntimeEnvironment` singleton object to license the app with a main license key and any extension licenses.

```
string licenseKey = "runtimelite,1000,rud#########,day-month-year,####################";
string[] extensions = {
"runtimeanalysis,1000,rud#########,day-month-year,####################" };

Esri.ArcGISRuntime.ArcGISRuntimeEnvironment.SetLicense(licenseKey, extensions);
```

Your app is now licensed to use the extensions.

Attribute Esri in your app

Esri requires that when you use an ArcGIS Online basemap, Esri data services, or Esri API technology in your app you must also include Esri attribution. There are specific requirements for attribution you may be required to address in your app depending on how your app is built and the data it uses. This is outlined in detail here in theAttribution in your app topic.

Copyright © 1995-2020 Esri. All rights reserved.

Deploy your app

Apps built with ArcGIS Runtime SDK for .NET require an ArcGIS Runtime deployment. The ArcGIS Runtime deployment contains the minimum set of ArcGIS Runtime components required to support the functionality in your app, from rendering beautiful maps to performing complex GIS analysis. Likewise, excluding unneeded components produces focused apps with a smaller footprint, containing bits only for the capabilities you need.

Define ArcGIS Runtime components to deploy

An ArcGIS Runtime deployment is created for you automatically when you build your project.

The ArcGIS Runtime SDK for .NET NuGet package(s) you add to your project will deploy the necessary runtime components according to your build configuration. For example, if you're building a WPF project for **Any CPU**, ArcGIS Runtime deployments will be created in directories called `client32` and `client64`. These directories appear under a folder called `ArcGISRuntime100.x` in the project output folder (`bin/Debug` or `bin/Release`, for example).

If you build your app for x86 or x64 (or choose **Prefer 32-bit**), the native DLLs will be placed in the same folder as your app's executable rather than in a separate `client32` or `client64` folder. You will still have an `ArcGISRuntime100.x` folder to contain some required resources.

You will need an ArcGIS Runtime deployment for every target platform for your app. The ArcGIS Runtime NuGet packages will automatically detect the build target for the project and build an ArcGIS Runtime with the appropriate architecture. This can be one or more of x86 (32 bit), x64 (64 bit), any CPU, or ARM depending on your app type.

Note:

Deployment steps may differ for each platform in **ArcGIS Runtime SDK for .NET** (WPF, UWP, Android, iOS). Consult the deployment information for each platform you want to deploy.

Deploying specialized symbols

You may need to deploy specialized symbols, such as military symbol dictionaries, for use with your ArcGIS Runtime SDK app. In ArcGIS Runtime SDK for .NET versions prior to 100.0, these symbols were included in the SDK install and could simply be

Copyright © 1995-2020 Esri. All rights reserved.

copied to a deployment folder for your app. To reduce the size of the setup (especially with the increased number of supported platforms), these symbols are not included in the ArcGIS Runtime SDK for .NET NuGet packages.

You have a couple of options for including these symbols in your deployment.

1. The ArcGIS for Defense group hosts the following ArcGIS Online items that contain .stylx files with military symbols. Use the support matrix to find and download the supported stylx file for your version of ArcGIS Runtime SDK. Once downloaded, you can include the files in your app deployment. Logic in your app, of course, would be required to locate and use the included files as needed.
2. Add logic to your app that prepares the app for offline use by downloading the files and storing them locally. You can access these resources when your user is online either by programmatically downloading them (using `System.Net.Http.HttpClient`, for example) or accessing them as portal items using the ArcGIS Runtime API, reading their contents, and storing them locally. The following example shows how to access some of the hosted military symbol items using ArcGIS Runtime code.

```
// Get the ArcGIS Online portal (default if no URI is provided)
ArcGISPortal agol = await ArcGISPortal.CreateAsync();

// Get the military symbol items (use their unique item IDs)
PortalItem itemMil2525c = await PortalItem.CreateAsync(agol, "ef95f95470db437f80ea764a9d05203b");
PortalItem itemMilStd2525d = await PortalItem.CreateAsync(agol,
"c78b149a1d52414682c86a5feeb13d30");

// Read the data from the items
var dataMil2525c = await itemMil2525c.GetDataAsync();
var dataMilStd2525d = await itemMilStd2525d.GetDataAsync();

// ... Code here to read the data and write it to a local file ...
```

Copyright © 1995-2020 Esri. All rights reserved.

Create a Local Server deployment

Local services allow you to deploy services that run locally with your app to provide functionality like geoprocessing, routing, and so on. In ArcGIS Runtime SDK for .NET, local services are only available for Windows Presentation Framework (WPF) apps. An app built for another ArcGIS Runtime for .NET platform can use such functionality from an online service, but not locally.

This topic does not apply for the iOS platform.

Copyright © 1995-2020 Esri. All rights reserved.

Reference

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.8.0

This page provides details about enhancements in the 100.8.0 release of ArcGIS Runtime SDK for .NET. It also lists this release's deprecations, resolved issues, and known issues. You can view release notes for past SDK versions by scrolling to the bottom of this page or by downloading past versions of this guide (in PDF format) from the downloads page (requires sign-in).

Enhancements

This section describes new features and improvements (what's new).

Maps and scenes

Image overlays in scene views

ArcGIS Runtime 100.8.0 introduces image overlays, which can be used to quickly render frequently changing images in a scene view. For example, rendering realtime sensor data, such as weather, where each static image represents a single frame from the radar. Image overlays render above all operational layers in the scene, but below any graphics overlays in the scene view. Designed for quick display, image overlays do not support the rich processing and rendering capabilities of a raster layer, which still provides the best option for workflows that require static image rendering, analysis, and persistence.

Online services supported in mobile map and scene packages

This release of ArcGIS Runtime now supports online services in mobile map and scene package files (.mmpk and .mspk). With ArcGIS Pro 2.6, users will be able to create packages that have a mix of local on-device content as well as online service content (such as weather or traffic information). If network connectivity is available, your app users can utilize online services, but if there is no connection, users can keep working with their local content. Apps that use packages that reference online services may need to handle authentication to ensure access to these services.

Relative to scene placement of features and graphics

Features and graphics can now be placed relative to objects in a scene, such as an elevation surface, buildings in a scene layer, integrated mesh layers, and extruded polygons. Note that some objects do not affect relative to scene placement (such as extruded points or lines, KML layers, or model symbols).

Improved text rendering in scene views

Text rendering in scene views has been greatly improved with a signed distance field (SDF) algorithm. In addition to being more efficient, and therefore using less resources, text in a scene view now supports background, outline, and halo colors, rotation relative to the screen or local north, strikethrough, underline, and better support for languages with connected characters (such as Bengali, Thai, or Arabic).

Support for visual variables for point features in 2D with dynamic rendering

To accurately represent visual variables, maintain consistency with static rendering mode, and enable developers to take advantage of the optimized display experience associated with dynamic rendering mode, ArcGIS Runtime 100.8.0 supports the use of visual variables for point feature layers in a map using dynamic rendering mode.

Copyright © 1995-2020 Esri. All rights reserved.

Arcade FeatureSet expressions in pop-ups

ArcGIS Runtime now supports pop-ups with Arcade expressions that use the `FeatureSet` capability. Feature sets allow you to create pop-up expressions that consume information from other layers in the map, from feature services, from portal items, or from related records. The new functions supported at this release are: `FeatureSetById`, `FeatureSetByName`, `FeatureSetByPortalItem`, and `FeatureSetByRelationshipName`. These expressions can be configured and saved in a web map and displayed in your ArcGIS Runtime apps. See the following blog for a description and example of using feature sets in your pop-ups: https://www.esri.com/arcgis-blog/products/arcgis-online/data-management/pump-up-your-pop-ups-with-arcade-and-the-living-atlas/

Geocoding

Support for 32-bit apps

The newer *.loz locators, introduced with ArcGIS Pro 2.3 and ArcGIS Runtime 100.5.0, are now supported for 32-bit apps.

Network analysis

Simulated location data source

A new simulated location data source allows you to mock incoming GPS positions using a polyline. This is can be used to test or demonstrate functionality that requires a device to be moving, such as navigating a route, when the device is not in motion. A location change event is raised for each point in the simulated data source and locations in the collection are visited in order, allowing you to simulate travel along a path at a defined rate of speed.

Directions engine improvements

The Directions engine in ArcGIS Runtime has been upgraded to match what is used in the rest of the ArcGIS platform. It is fully backwards compatible with existing transportation network datasets, requires no API changes, and provides results that are consistent with other ArcGIS products such as ArcGIS Pro, Server, and Online. Furthermore, the performance of generating routes and turn-by-turn driving directions has increased by an average of 50% on desktop machines, and by 30% on mobile devices.

Raster

Identify raster layers

New to this release of the ArcGIS Runtime is the ability to perform identify operations on raster layers to obtain individual raster cell (or pixel) values. Identify is supported for raster layers referencing ArcGIS Image Services, or local raster files (on disk or device). The raster cell values can be displayed in well-formatted UI pop-ups created by the developer or from pop-ups defined in web maps and web scenes authored in ArcGIS Online or ArcGIS Enterprise.

Utility network

Isolation trace

A new trace type, called the isolation trace, has been added. Isolation traces are used to determine the minimum set of operable features required to stop a network's resource, effectively isolating an area of the network. For instance, when a leak occurs on a water network, a manager can use the isolation trace to identify the valves that must be closed to eliminate water flow at the leak location.

Copyright © 1995-2020 Esri. All rights reserved.

Trace filters

Trace filters are a new configuration option for utility networks that restrict the results returned by a trace without preventing it from traversing to the source. These filters allow utility managers to examine the nearest upstream protective devices.

Visualize utility network associations

New with ArcGIS Runtime 100.8 is the ability to visualize network associations on a map using a geometry that represents structural attachment and connectivity associations.

As ArcGIS Runtime continues to add utility network functionality with the 100.8.0 release, the following describes some of the capabilities that are not currently supported.

- You cannot take the utility network offline. However, you can take simple features (that are in the utility network) offline and edit them with Collector for ArcGIS or other ArcGIS Runtime apps.
- Editing capabilities are limited. You can edit utility network features as simple features, but you cannot perform utility-network-specific edits such as creating or removing associations. Attribute rules will run on the server, and dirty areas will be created, but there are no capabilities to validate network topology from ArcGIS Runtime.
- You can connect only to the default version of the feature service.

Licensing enhancements

You no longer need a Utility Network extension license to load or display a `UtilityNetwork` object. The license is needed only to perform traces.

Miscellaneous

- A consistent error code for ArcGISRuntimeExceptions is now thrown for null arguments. In previous releases, in cases where a core ArcGISRuntimeException was thrown to report an unsupported null argument, the error code may have been either 1 (null pointer exception) or 2 (invalid argument exception). In this release, error code 1 (null pointer exception) will be reported consistently in this situation. For additional information on error codes, see this guide's error code topic.

API changes

Behavior changes

- In some cases, for example some null object errors and parameter out of range errors, the **additional information** message returned from `ArcGISRuntimeException` may use a generic term such as `object` instead of the specific parameter name resulting in the error.
- The `isSketchValid` members of `SketchEditor` and `GeometryBuilder` now check for a minimum number of points in each existing part of a multipart geometry. A polyline must contain at least two valid points in each part, and at least one part. A polygon must contain at least three valid points in each part, and at least one part. Prior to version 100.8.0, only one part was required to have the minimum number of points. There are no changes for other geometry types.

Deprecations

Support for iOS 11 was deprecated in the previous release and ArcGIS Runtime SDK 100.7.0 was the last release to support it. A minimum of iOS 12 is required at 100.8.0.

Copyright © 1995-2020 Esri. All rights reserved.

The classic geocode locator format (*.loc) will not be supported after ArcGIS Runtime version 100.9.0. After that release, only the newer *.loz locators, introduced with ArcGIS Pro 2.3 and ArcGIS Runtime 100.5.0, will be supported.

Issues resolved

- BUG-000129571 ArcGIS Earth crashes on adding layer from ArcGIS online when the symbology property for a 3D web layer is set to a Procedural Fill.
- BUG-000119455 GeocodeResult.Label property returns no value for some geocoded addresses.
- ENH-000114198 Provide a functionality to retrieve Raster Information through identify operation.
- BUG-000121574 The downloaded vector tile package does not contain any attribution.
- BUG-000126426 MapView.identifyLayerAsync does not return results from a map service containing a large number of layers.
- BUG-000125810 Changing the FeatureLayer.ScaleSymbols property when the feature layer is a point layer and part of a GroupLayer causes a crash.
- BUG-000128841 HaloColor and HaloWidth properties of the Text symbol are not displayed correctly in SceneView.
- BUG-000128759 Graphics flicker on MapView when the mouse moves over the view.
- ENH-000124541 Provide functionality to retrieve raster cell values through identify or query operation.
- BUG-000113124 Secondary raster will not appear when added to mosaic dataset that is currently being displayed.
- ENH-000126129 While defining LabelExpression for Symbol using JSON, properties like borderLineColor and borderSize do not work as expected for FeatureLayers/Graphic Overlays.
- BUG-000129806 Labels flicker when identifying graphics on the map.
- BUG-000129488 A line graphic is still visible when a point is removed from the Point Collection for specific line symbols GFGPGLB and GFGPGAG in mil2525c style.
- ENH-000125141 Provide upport for fast refreshing of raster data.
- GeoNet Post: FeatureLayer.Clone drops DefinitionExpression, Renderer, SelectionColor, SelectionWidth.
- BUG-000122974 ShowCalloutForGeoElement method in SceneView Class causes the application to crash when the background of the identified graphic is not on the earth surface.
- BUG-000128886 An exception is thrown when several Windows Form windows with embedded MapControls are opened.
- BUG-000128196 Clearing or removing the GraphicsOverlay does not cause applications to recover the memory allocated to load the graphics.
- BUG-000122234 Finger-tapping a button inside a CalloutDefintion fails to pop up a message box.
- BUG-000118353 OAuth workflow fails for when authenticating a WebMap containing layers that are added to ArcGIS Portal as item with stored credentials.

Known issues

- BUG-000127369 ArcGIS Runtime applications do not display label callout angle in a mobile map package (.mmpk).
- BUG-000126599 Feature symbology fails to display halos.
- BUG-000127371 ArcGIS Runtime applications fail to display raster symbology in a mobile map package (.mmpk).
- BUG-000127860 Labels fail to display when querying the esriFieldTypeDate field in definitionExpression.

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000129458 Large vector tile package (vtpk) crashes in x86 Runtime application while zooming or panning
- BUG-000128862 Slow layer rendering performance from a local mobile geodatabase containg large amounts of data.
- BUG-000122413 Querying a not nullable shapefile field ("FID") in the WhereClause causes application to crash.
- BUG-000126301 A tile package (.tpkx) created using the Export Map Server Cache tool fails to load
- BUG-000127930 Panning and zooming in the Xamarin.Forms iOS app causes a crash on when running on an iPad.
- BUG-000123820 The display freezes when minimizing or maximizing the ArcGIS Runtime UWP and Xamarin Forms application with navigation enabled.

Related topics

System requirements
Install the SDK
Develop your first map app
Release notes for 100.7.0
Release notes for 100.6.0
Release notes for 100.5.0
Release notes for 100.4.0
Release notes for 100.3.0
Release notes for 100.2.0
Release notes for 100.1.0
Release notes for 100.0.0

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.7.0

This page provides details about enhancements in the 100.7.0 release of ArcGIS Runtime SDK for .NET. It also lists this release's deprecations, resolved issues, and known issues. You can view release notes for past SDK versions by scrolling to the bottom of this page or by downloading past versions of this guide (in PDF format) from the downloads page (requires sign-in).

.NET Core

This release adds support for building Windows Desktop applications using WPF with .NET Core 3.1. This marks a significant step for the popular WPF UI framework by liberating it from the system-wide .NET Framework on which it currently depends, enabling you to deploy smaller, self-contained apps that may target different .NET Core versions while continuing to benefit from future enhancements beyond .NET Framework. To get started building Windows Desktop apps on .NET Core you need Visual Studio 2019 16.4 (or later) with the workloads **.NET desktop development** and **.NET Core cross-platform development**. ArcGIS Runtime for .NET also provides a project template, "ArcGIS Runtime App (WPF .Net Core)". For more information see .NET Core 3 for Windows Desktop.

Enhancements

This section describes new features and improvements (what's new).

Utility network

Utility network functionality has been greatly enhanced in this release. You can now trace your electric circuits, and gas and water systems using **upstream, downstream and subnetwork traces**. Set numerous trace configuration options to enhance and fine-tune all of these traces. In addition, your users can now query the contents of a network container, such as a switch cabinet or regulator station, using association queries. Finally, using the new **subtype feature layer**, you can now efficiently display different asset groups within a single feature class.

Maps and scenes

New raster types as offline elevation sources

With ArcGIS Pro 2.5 you can create mobile scene packages that **utilize a range of raster types as elevation sources** within a scene. Use these packages to take your scenes offline. With this release you do not need to unpack packages that contain raster data.

Custom styles available with vector tiles in a mobile map package

With ArcGIS Pro 2.5 you can create mobile map packages that include vector tile layers along with their own custom style. This release of ArcGIS Runtime adds support for **displaying vector tile layers using the custom style** included in the package.

Offline routing with a scene

With ArcGIS Pro 2.5 you can create mobile scene packages that contain transportation networks. You can now use these packages with ArcGIS Runtime to perform **routing analysis entirely offline in a scene**.

Map background color

A map's **background color is now honored** when displaying maps in a map view. This color will be displayed in transparent areas of the map as well as areas where there is no basemap or operational data. The background color can be defined programmatically or specified while creating webmaps and mobile map packages.

Copyright © 1995-2020 Esri. All rights reserved.

Drape 2D markers on a surface

Rather than always being billboarded, 2D markers can now be **draped on the ground on a 3D surface** when drawn in dynamic rendering mode. This matches the existing behavior of lines and polygons.

MrSID performance improvement

Display times are considerably faster when initially displaying and interacting with MrSID raster datasets at this release.

Closest Facility and Service Area inputs

Additional properties have been added to the `Facility` and `Incident` classes to help **model and accurately locate moving features** used as inputs. These properties are used to describe the speed, bearing, bearing tolerance, and latency for an input.

Feature table performance improvements

Performance of `FeatureCollectionTable`, `ServiceFeatureTable`, and `WfsFeatureTable` has been improved, in some cases by up to 65%. This applies to the initial population and subsequent navigation for `ServiceFeatureTable` and `WfsFeatureTable` when using **On Interaction Cache** or **On Interaction No Cache** feature request modes, as well as the manual population of these tables if using **Manual Cache** mode with calls to `PopulateFromServiceAsync`. It also applies to the insertion of large numbers of features into `FeatureCollectionTable`.

Symbology

Model scene symbol support for glTF

Model scene symbols now support glTF 2.0, a standardized file format for describing 3D objects. Model symbols can be created from .glTF or .glb files (a binary form). See the glTF Overview for more information about the glTF 2.0 specification.

Support for v3.0 dictionary styles

This release of ArcGIS Runtime adds support for version 3.0 of the ArcGIS dictionary style. This is the latest version used by ArcGIS Pro 2.5.

Visible scale range for symbols

ArcGIS Runtime now **honors the visible scale range for symbols** when displaying maps from mobile map packages. This is useful to limit the amount and type of data displayed at different map scales. You can use ArcGIS Pro to define the scale range for symbols in a Unique Value or Class Breaks renderer of a feature layer.

Improved support for visual variables with 3D Symbols

Model scene symbols now support size and rotation from visual variables along a user-defined axis. Geographic and arithmetic rotation from visual variables is supported along the heading axis. Color and size from visual variables are now supported on 3D primitive marker symbols, such as cube, cylinder, etc.

Copyright © 1995-2020 Esri. All rights reserved.

Create and modify KML screen overlays

Previous releases introduced the creation and modification of certain types of KML content, such as placemarks and ground overlays. This release adds support for **creating, editing, and saving screen overlays** (`ScreenOverlay`) within KML datasets.

API additions comprise:

- A constructor for `ScreenOverlay`
- Read-write properties for color, icon, and draw order on `ScreenOverlay`
- New properties on `ScreenOverlay` to support positioning, sizing, and rotating screen overlays

Augmented reality

The AR Toolkit now **allows you to clip the contents of a scene** by setting the `clippingDistance` property. This allows you to build more realistic table-top experiences where you can limit the 3D content being displayed to the surface upon which it is placed, and also helps improve the performance of full-scale experiences by prioritizing the content that is nearby over data that may be far away.

Licensing enhancements

With ArcGIS Runtime 100.7, you can now **unlock Standard** and **Advanced** license levels and the **Analysis** extension when using the **Named User** licensing mechanism with ArcGIS Online or ArcGIS Enterprise 10.8 or higher. Previously these license entitlements were only available using a license key.

API changes

Behavior changes

- Utility Network license extension—Starting with ArcGIS Runtime 100.7 and ArcGIS Enterprise 10.8, Utility Network capabilities now require an explicit **Utility Network** license extension. This extension can be unlocked using a license key, or a named user that includes the Utility Network user type extension.
- When you take a webmap offline you can navigate between the map's min and max scale. This scale range is now **specified by the original webmap** (either defined by the service or the basemap layer). Previously, this scale range was limited by levels of detail downloaded when taking the map offline.
- When building an app that uses the WPF NuGet package and not building for **AnyCPU** (prefer 32-bit or x86/x64, in other words), the native DLLs were previously copied to the arcgisruntime100.x/clientXX folders. With 100.7.0, they are now placed in the same folder as the executable.

Deprecations

Support for iOS 11 is deprecated. A minimum of iOS 12 will be required at the next release.

Support for Android API level 19 (Android 4.4, KitKat) is deprecated. A minimum of API level 21 (Android 5.0, Lollipop) will be required at the next release.

Support for Windows 7 SP1 is deprecated. A minimum of Windows 10 version 1803 will be required at the next release.

Support for Windows 10 version 1709 is deprecated. A minimum of Windows 10 version 1803 will be required at the next release.

Copyright © 1995-2020 Esri. All rights reserved.

In addition to being billboarded, 2D marker symbols displayed in a scene can now be draped on the ground to match the behavior of lines and polygons. This is supported for dynamic rendering, allowing you to move and rotate the symbols at sub-second intervals. To support this change, the `Draped` enum value of `SurfacePlacement` is deprecated. Use either `DrapedBillboarded` or `DrapedFlat` instead.

The `PortalInfo.licenseInfo` property has been deprecated. The preferred way to obtain license information for a Named User is by using the new `Portal.fetchLicenseInfo()` method. This method returns ArcGIS Runtime license entitlements of Named Users in ArcGIS Online and ArcGIS Enterprise 10.8, and is backwards-compatible with older versions of ArcGIS Enterprise.

Raster data can now be **read directly from a mobile map/scene package** without the need for it to be unpacked. You do not need to check whether a package can be read directly. The `IsDirectReadSupportedAsync` method on the `MobileMapPackage` and `MobileScenePackage` have been deprecated. From this version they will always return a value of true. Since this method is no longer required, it can be removed from your calling code along with any subsequent use of `UnpackAsync` method.

Issues resolved

- ENH-000124453 Provide the ability to display a mobile map package and respect the visible scale range set for each symbol class in ArcGIS Pro for that mobile map package.
- BUG-000126697 Application crashes when attempting to load a certain mobile map package (.mmpk).
- Callout appears "collapsed" if it was initially hidden.
- BUG-000126144 Crash in Map_renderer::Labelable.
- BUG-000124577 Poor performance with large MrSID rasters.
- ENH-000115312: Clarify and improve the documentation for requirement and approaches for offline preplanned workflow.
- BUG-000120409 WMTS service does not load if the tile matrices of the service in the GetCapabilities file are not in sequential order.
- BUG-000109943 When creating the points, if the coordinate value is lower than decimal point third place, the raster is added to the map with an unexpected rendering result.
- BUG-000122510 Selecting features on a feature layer using featureLayer.SelectFeaturesAsync() uses the geometry of the polygon extent rather than the geometry of the polygon itself when querying from an event listener.
- BUG-000115811 Blend renderer for scene view renders in different color range than for map view.
- ENH-000120897 Support draped mode for symbols in dynamic mode 3D.
- BUG-000126415 Application crashes when rendering labels with query with the EXC_BAD_ACCESS error
- BUG-000126467 Display fails for online and synced offline annotation layers with certain spatial reference characteristics
- BUG-000125578 The `LabelDefinition.ToJson()` method returns an empty symbol value.
- Regression: Refresh token is being exchanged when an access token refresh is required. See GeoNet post.
- BUG-000123380 CallOutDefinitionClass is unable to display the message box when the camera pitch angle is more than 100 degrees.

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000109177 Xamarin.iOS Storyboard Error: "Custom Components are not being rendered because problems were detected"

Known issues

- BUG-000127070 The `WGS 1984 (Transit)` spatial reference (WKID 104016) is no longer recognized, and data or services using this WKID may fail to load. As a workaround, the EPSG replacement code `8888` can be used instead.
- BUG-000125987 The watermark is not removed without rotating the screen after setting the license from a named user login.
- BUG-000125249 A small outline added to graphics when NavigationConstraint is set to none.

Related topics

Release notes
Release notes for 100.6.0
Release notes for 100.5.0
Release notes for 100.4.0
Release notes for 100.3.0
Release notes for 100.2.0
Release notes for 100.1.0
Release notes for 100.0.0

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.6.0

This page provides details about enhancements in the 100.6.0 release of ArcGIS Runtime SDK for .NET. It also lists this release's deprecations, resolved issues, and known issues. You can view release notes for past SDK versions by scrolling to the bottom of this page or by downloading past versions of this guide (in PDF format) from the downloads page (requires sign-in).

Enhancements

This section describes new features and improvements.

Utility network

You can now use an utility network with ArcGIS Runtime. Utility networks provide a comprehensive functional framework in ArcGIS for modeling utility systems such as electric, gas, water, storm water, wastewater, and telecommunications. It is designed to model all of the components that make up your system—such as wires, pipes, valves, zones, devices, and circuits—and allows you to build real-world behavior into the features you model. The utility network framework enables advanced out-of-the-box analytics and visualization for geographic location, network topology, or complex assemblies. For details about the capabilities included in 100.6, see Analyze a utility network.

Route navigation

ArcGIS Runtime SDK can now navigate a route between two or more locations across a transportation network. The new navigation API gives you the ability to enhance the routing experience by using the current device location to track progress, and can provide navigation instructions (maneuvers) as the user travels the route. You can integrate driving directions with your device's text-to-speech capability and automatically re-calculate a new route when the user leaves the current one. Implementing navigation in your runtime app requires a Basic license. For more details, see Navigate a route.

Maps and scenes

Preplanned scheduled updates

This release introduces an optimization for the preplanned offline workflow using **scheduled updates**. In this scenario, web map authors can configure map areas to periodically update feature data on a regular schedule and stage those changes on the portal for distribution. Runtime apps that have taken those map areas offline can then simply check for updates and download the changes directly from the portal without having to perform expensive sync operations with the backing feature services. This workflow increases the scalability of working with offline maps since all users can retrieve updates to their map areas using the same set of downloads that are available from the portal without increasing load on backing feature services.

Note:
This optimization is only applicable if your app makes read-only use of offline map areas. If your app needs to perform offline editing of feature data in the map area, you should continue to utilize the existing preplanned workflow which syncs directly with feature services.

Close mobile map/scene packages

A new `close()` method has been added to Mobile Map Package and Mobile Scene Package to **force close** connections to the package. This releases any file locks for the data on disk thereby allowing apps to remove or change the underlying package file or directory. This is particularly helpful for garbage collected platforms where simply releasing references to the package

Copyright © 1995-2020 Esri. All rights reserved.

object in the app does not guarantee that the object will be destroyed and any file handles it has will be closed. After closing the package, do not continue to use any objects (maps, scenes, layers, tasks, features, etc) that were retrieved from the package

Improved performance

Performance of Mobile map packages containing a large number of layers such as those that include Streetmap Premium data has been improved. In some cases the time taken to display the initial map data is **3 times as fast** and uses **33% less memory** as compared to the previous version.

Group layers in mobile map packages

Mobile map packages (`.mmpk`) now preserve group layers that are defined in a map authored in ArcGIS Pro. Group layers are no longer flattened into individual layers, and the group layer is visible in the layer list. Overrides affecting min/max scale, opacity, and visibility properties of a group layer are honored in a mobile map package. Support for group layers in web scenes and web scene packages was introduced in an earlier release; the current release brings full group layer support to mobile map packages as well.

Enhanced dictionary symbology

ArcGIS Runtime SDK now supports a new format of dictionary style file introduced by ArcGIS Pro version 2.4. This format uses Arcade for the logic to generate symbols and is therefore fully customizable. Use of the older format styles is supported but deprecated in this release. For more information about using custom dictionary symbols in ArcGIS Runtime, see the Display symbols from a style with a dictionary renderer topic in the developer guide. To create a custom dictionary style, refer to the Dictionary renderer toolkit project on GitHub.

Create and modify KML content

Previous releases supported reading and displaying KML content (`.kml` and `.kmz` files) from your local disk and from the web. This release enables you to **create** new KML content, or **modify** existing content. The following KML content types are supported for creating, editing, and saving to KML datasets:

- Placemarks (point, line, polygon)
- GroundOverlays
- Folder and document containers
- Network link parameters

Note:
Creating and editing a KML file requires a Standard level license.

Exceptions to editing KML files

The following KML content types currently cannot be created, edited, or saved:

- ScreenOverlays
- Shared styles (Geometry, Placemarks, Overlays)
- Highlight Icon styles
- KML datasets with multiple root nodes

Copyright © 1995-2020 Esri. All rights reserved.

Custom parameters for WMTS and WFS requests

ArcGIS Runtime SDK now supports custom parameters for WMTS and WFS requests. Custom parameter properties now appear on the `WmtsService`, `WmtsLayer`, and `WfsService` classes. This support is consistent with ArcGIS Runtime's existing support for custom parameters on WMS requests in the `WmsService` class. Custom parameters are appended as key/value pairs to WMS, WMTS, and WFS requests such as `GetCapabilities` and `GetTile`. Setting custom parameters allow you to specify additional parameters to service request such as API keys or user credentials. See the API documentation of these classes for details.

3D layers

Optimized scene layer loading pattern for desktop

Devices with adequate memory resources (such as desktops) employ a new scene layer loading pattern that provides more intuitive draw and navigation performance. Scene layers load and render more quickly with content nearest the camera loaded first and at the highest appropriate resolution. Initially, a coarse representation covering the full extent is loaded. This allows the camera to spin around and not see empty spaces. Finer detail is added progressively. Constrained-memory devices, such as mobile devices, are not affected by this enhancement.

Elevation offset

A feature layer or a graphics overlay in your ArcGIS Runtime application can now support elevation offsets defined in a scene defined in ArcGIS Pro or the Scene Viewer. You can also read and modify the elevation offsets.

The`AltitudeOffset` property affects all features in a feature layer or all graphics in a graphics overlay. Classes that now have the property are:

- `ArcGISSceneLayer`
- `LayerSceneProperties`
- `IntegratedMeshLayer`
- `PointCloudLayer`

The altitude offset property works with web scenes or scenes from a mobile scene package (.mspk). Units are in meters.

Point scene layers

Point scene layers in your ArcGIS Runtime applications now have additional functionality: your users can select a point and identify its attributes in a popup. Point scene layers provide fast, responsive display of point features in a 3D view based on scale, distance, and threshold parameters associated with a viewpoint. Point scene layers are generated from point feature layers and can be loaded from a web scene, mobile scene package, or directly from a service. Note that when a scene is shared as a web scene in ArcGIS Pro, any point feature layers present in the 3D Layers section of the Contents pane will be published as a scene service. Thus, publishing a point feature layer in a web scene automatically generates a point scene layer.

Supported 3D file formats

Supported 3D file formats are as follows:

- `*.dae` - Collada
- `*.obj` - WaveFront

Copyright © 1995-2020 Esri. All rights reserved.

- `*.3ds` - 3D Studios
- `*.fbx` - Filmbox

For a list of 3D file formats that were mentioned in previous releases but are now deprecated, see Deprecations section.

Ukrainian language support

Localized Directions returned by route task and closest facility task are now available in Ukrainian.

Annotation

Runtime users can now take annotation data offline from a sync-enabled feature service. They can create annotation layers from ArcGIS feature tables contained in either a geodatabase or a feature service. Annotation features within an annotation layer are partitioned into annotation sublayers. Accessing the sublayers allows Runtime users to read the published metadata and manually toggle visibility of the sublayer.

Behavior changes

- Applying a viewpoint constructed with a **map scale now displays a different visible area in a scene view**. In previous versions, a much larger scale value had to be provided to create a visible extent comparable to the same scale applied to a map view. A viewpoint constructed with a scale value now displays roughly the same area when applied to a map view or a scene view.
- Runtime will now **use a default renderer when displaying various raster formats**. This change improves display consistency when displaying raster data if a renderer is not specified.
- Selection for dictionary symbols works slightly differently at 100.6 than it did at 100.5. **If a dictionary symbol has text, the text is now selected along with the rest of the symbol**. This is a behavior change for existing Runtime users, and is similar to selection in ArcGIS Pro.

Deprecations

Support for Windows 7 is deprecated. The last release to support Windows 7 will be version 100.7.

Support for Windows 10 version 1703 is deprecated. With the 100.7 release, the minimum will be Windows 10 version 1709.

Some 3D file formats used for 3D symbols are deprecated. The last release to support these formats is 100.6.0.

- Drawing Exchange Format `*.dxf` files
- Direct X `*.x` files
- Polygon file format `*.ply` files (Standard polygon library)
- Stereo lithography CAD `*.stl` files
- Industry foundation classes `*.ifc` files
- Blender `*.blend` files

Issues resolved

- BUG-000123552 An extruded line graphic fails to render in a scene view when exceeds a certain height.

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000123647 An access violation occurs in SceneView when enabling labels on graphics overlays..
- BUG-000121670 Graphics overlays with the visibility turned off are flashing with around 10 or more graphics overlay layers.
- BUG-000121802 Application crashes if the path location contains a Chinese character.
- Access violation calling SuggestAsync on a composite locator from a mobile map package. For more information see the GeoNet post, Access Violation In Runtime 100.5.
- BUG-000123182 A point layer from a local geodatabase (GeodatabaseFeatureTable) fails to display when sharing the same ArcGISMap between two activities.
- BUG-000121882 Wrapping the MapView control with the ScrollViewer control causes loss of ability to pinch-zoom the map
- ENH-000118206 The ArcGIS Runtime documentation is unclear about how to use Integrated Windows Authentication in an ArcGIS Runtime Application
- BUG-000116894 The documentation states that the MapView.SetViewpointGeometryAsync method will accept a point as the geometry, but doing so returns the error "Invalid argument: target_extent has zero for width or height."

Known issues

- BUG-000122050 The QueryFeaturesAsync method in ServiceFeatureTable class does not request M values for features.
- BUG-000122510 Selecting features on a feature layer using featureLayer.SelectFeaturesAsync() uses the geometry of the polygon extent rather than the geometry of the polygon itself when querying from an event listener.
- BUG-000122389 LocalMapService.StartAsync throws an exception when a path to the map package contains spaces.
- BUG-000121666 Creating features with PolygonBuilder with projected coordinate system (for example, with WebMercator) does not work as expected in a SceneView.
- BUG-000121134 Feature layer labels render differently depending on whether the layer is part of the web map or not.
- BUG-000120751 A scene layer package created from OSGB data in ArcGIS Pro by the Create Integrated Mesh Scene Layer Package tool fails to display.
- BUG-000120229 The ArcGIS Online web map feature does not sync to the preplanned area when the feature is moved out of the extent of the preplanned map area.
- BUG-000120165 MapView.SetViewpointAsync() returns a larger extent than specified.
- BUG-000119567 CountryCode and Category properties in the SuggestionParameters function do not work with the LocatorTask function from a mobile map package.
- BUG-000117828 The performance of Surface.GetElevationAsync() is not optimal when invoked initially. It improves with each invocation.
- BUG-000117556 Some graphics overlay labels may not always be displayed.
- BUG-000117177 Rendering extrusion for line features fails to work as expected in a 3D scene.
- BUG-000117102 Feature class used to create locator that has M-value does not populate with the value of Match_addr and Label.
- BUG-000116969 Setting SceneSymbolAnchorPosition.Origin does not work when the 3D model has an offset.
- BUG-000116792 The layers in a mobile map package created with ArcGIS Pro do not respect label stacking properties.
- BUG-000116356 Loading some mobile map packages may cause a crash.
- BUG-000116322 When consuming a vector tile package, the offset property of symbol layers is not honored.

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000116235 The SelectFeaturesAsync and QueryFeaturesAsync methods return more than one result even when tolerance is set to 0.
- BUG-000116234 The GetElevationAsync method returns a different elevation from a raster when compared to the Get Cell Value method in ArcMap.
- BUG-000116209 Labeling feature weights assigned to a feature class in ArcGIS Pro 2.2 mobile map package is not honored.
- BUG-000116022 Downloading the map area returns the error message "Unable to complete operation" when the web map contains an ArcGIS Server feature service using versioned data.
- BUG-000115855 Geometry does not return the M-value even if the Geometry.HasM property is true.
- BUG-000115750 Tile packages created from certain tiling schemes neither display nor provide any exception.
- BUG-000115711 ArcGISFeatureTable.QueryRelatedFeaturesAsync fails to return the related records data in some cases.
- BUG-000114021 Trying to delete a raster used by a raster layer causes an error.
- BUG-000110655 UniqueValueRenderer does not properly render a dynamic SublayerSource of a LocalMapService.
- BUG-000123710 The SceneView.ScreenToBaseSurface Method sometimes causes the application to freeze.
- BUG-000120490 The Raster To Polygon geoprocessing package (.gpk) fails to start as a local geoprocessing service when created from ArcMap 10.6.1 packaged from a geoprocessing tool result.
- BUG-000121574 A downloaded vector tile package does not contain any attribution.
- BUG-000121695 MapView.IdentifyLayersAsync applied to a layer with a DefinitionExpression results in poor performance.
- BUG-000115811 Blend renderer for scene view renders in a different color range than for map view.
- BUG-000123744 When a shapefile is added to a Local Server dynamic workspace, 32-bit Microsoft Excel launches 10 seconds slower.
- BUG-000115068 - The LegendInfo of Local Server based ArcGISMapImageLayer sublayers created from shapefiles is not updated to represent sublayer renderer symbology
- BUG-000121762 OpenStreetMap Vector Basemap displays large gaps between dashed lines for trail polylines on a high resolution display.
- BUG-000122693 Features from large shapefiles fail to render when the ArcGISMapImageSublayer.setLabelsEnabled() parameter is set to true.
- BUG-000120169 Large raster files (.tif) sometimes fail to load.
- BUG-000122234 Finger-tapping a button inside a CalloutDefintion fails to pop up a message box.
- BUG-000122054 Xamarin iOS application crashes inconsistently with "System.Device.Location.GeoCoordinate.set_Course: The value of the parameter must be from 0.0 to 360.0".
- BUG-000118161 Setting 'MapView.LocationDisplay.IsEnabled = true' returns the error message, "already set to true" when the device network connection is slow.
- BUG-000120895 AuthenticationManager.Current.GenerateCredentialAsync causes the Windows Form to reduce in size.
- BUG-000119951 The "Sketch graphics on the map" sample for Xamarin Android crashes when adding many vertices to a polygon, polyline, and multipoint.
- BUG-000119636 Local raster layer disappears when zooming out in the Xamarin Universal Windows Platform (UWP) application.

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000118500 The error message, "Invalid argument: location data source has been started" is returned when setting MapView.LocationDisplay.IsEnabled to true after multiple calls when the device has a slow internet connection.
- BUG-000118353 OAuth workflow fails when authenticating a web map containing layers that are added to Portal for ArcGIs as items with stored credentials.
- BUG-000117872 The view's GeoViewDoubleTapped event is registered with two GeoViewTapped events when the view interaction options (MapViewInteractionOptions/SceneViewInteractionOptions) are disabled.
- BUG-000114932 When the map view is zoomed or panned, labels do not reposition to move in view as they do in ArcGIS Pro.

Related topics

Release notes
Release notes for 100.7.0
Release notes for 100.5.0
Release notes for 100.4.0
Release notes for 100.3.0
Release notes for 100.2.0
Release notes for 100.1.0
Release notes for 100.0.0

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.5.0

This page provides details about enhancements in the 100.5.0 release of ArcGIS Runtime SDK for .NET. It also lists this release's deprecations, resolved issues, and known issues. You can view release notes for past SDK versions by downloading past versions of this guide (in PDF format) from the downloads page (requires sign-in).

Enhancements

This section describes new features and improvements.

Maps and scenes

- **Mobile scene packages** are now supported in the ArcGIS Runtime allowing you to take scenes and their associated data offline. You can consolidate scenes and their data into a single `*.mspk` file using ArcGIS Pro 2.3 or higher and share the package with your ArcGIS organizational account or copy it directly to your mobile device.
- Maps can now be rendered with support for **reference scale**. Reference scale is the scale at which symbols and labels appear at their intended, true size. As you zoom in and out beyond the reference scale, symbols and text will increase or decrease in size relative to the reference scale to keep a consistent size on the map. If no reference scale is set, symbols remain a constant size on the screen and do not change size as you zoom in or out. Individual layers in a map can opt in or out of honoring the map's reference scale.
- **Annotation layers** are now supported for display in maps. Annotation allows you to precisely control the placement and layout of text and is often used as an alternative to labels which are more dynamic. Annotation layers can be included for offline use in mobile map packages created from ArcGIS Pro 2.3 or higher or accessed over the network through Feature Services from ArcGIS Enterprise 10.7 or higher.
- **Group layers** are now available. Group layers combine multiple layers that are often displayed and managed together. You can create group layers locally from scratch in maps and scenes. Currently, saving group layers is only supported in web scenes.
- **Point Scene layers** are now supported to provide fast visualization of large amounts of 3D point features from ArcGIS Scene Services, Mobile Scene Packages (.mspk file) or Scene layer packages (.slpk file). Point scene layers use automatic thinning to improve performance by not displaying all the features at greater distances, rather, as you zoom in, additional features are displayed until you reach the highest level of detail, when all features will be shown.
- **Point Cloud layers** are available to support the display of sensor data such as LiDAR from ArcGIS Scene Services, Mobile Scene Packages (.mspk file) or Scene layer packages (.slpk file).
- **Displaying subsurface** content is now supported in scenes allowing you navigate that camera below the terrain and explore the data underneath. You can change `Surface.navigationConstraint` property to allow navigating below ground or you can change the opacity of the surface to view subsurface content from above the ground.

KML Tours

ArcGIS Runtime now supports **playing tours** contained in a KML file. Tours are a great way to guide users through KML data and highlight important aspects through narration and animations. The following tour actions are supported -

- `FlyTo` - fly to a specified location using a bounce or a smooth flying mode
- `Wait` - wait the specified length of time before executing the next tour primitive

Copyright © 1995-2020 Esri. All rights reserved.

- `AnimatedUpdate` - update KML features. Any changes that lend themselves to interpolation (that is, that contain intermediate states, such as changes in size) are animated over the specified duration.
- `SoundCue` - play the specified sound file. Playing multiple sound files in parallel are supported.

See topic Play KML tours for more information.

Web Feature Service (WFS)

ArcGIS Runtime now supports **OGC Web Feature Service**, including versions 2.0.0 and 2.0.2. You can:

- Connect to a WFS service and discover the available datasets
- Query data from a WFS service and display in a 2D map or 3D scene
- Query and load non-spatial datasets
- Use XML to perform advanced queries on the server

At 100.5.0, Runtime does not support editing WFS, taking WFS offline, opening WFS from a portal item, or reading and writing WFS from web maps and scenes. Runtime does not support complex features (those with nested attributes), which may affect INSPIRE services. Some geometry types, including those with Z and those with M, are not supported. When loading a WFS feature table, fields with names that start with **gdb_** will be ignored. When populating a WFS feature table from a service, the QueryParameters `WhereClause` is only supported on services that accept **CQL_FILTER**, which includes GeoServer.

Symbology

- New **Multi-layer Symbols**API offers more flexibility and control over how to construct and represent symbols. These symbols can be composed of a number of different symbol layers that are combined together along with geometric effects to achieve high cartographic quality. The API closely mirrors the definition of symbols in the Cartographic Information Model spec.
- **Arcade expressions** on renderers in a web map or mobile map are now honored in ArcGIS Runtime. Expressions can be used in place of raw field values on these renderers to support smart mapping styles such as Predominance, Relationships, etc. These expressions can be authored using ArcGIS Pro or the Map Viewer web application in ArcGIS Enterprise or ArcGIS Online.
- 3D Line symbols such as **volumetric tubes** are now supported when displaying feature layers containing polyline geometry in web scenes.

Offline

- You can now choose to **use an existing basemap** that is located on the device when taking a map offline. This feature, called by reference basemap, removes the overhead of downloading a basemap for every map and is available for both the on-demand and preplanned offline workflows. The path to the basemap can be an absolute path or a path relative to the mobile map package.
- Image and vector tiles can now be downloaded from ArcGIS Online services using a **polygon as the area of interest**. This gives you greater precision to request only the tiles you need to support your operational data. This polygon support is available through `ExportTileCacheTask`, `ExportVectorTilesTask` and `OfflineMapTask`. ArcGIS Enterprise services already supported polygonal areas of interest with prior versions of Runtime.
- The new **Compact Cache V2** format of tile packages (`*.tpkx` file) available with ArcGIS Pro 2.3 is now supported in the ArcGIS Runtime. This format is based on an open specification and is optimized for fast access and improved performance.

Copyright © 1995-2020 Esri. All rights reserved.

You can use the existing `ArcGISTiledLayer` and `TileCache` APIs to support the new format. See Take a layer offline topic for more information on tile packages.

- The `ExportTileCacheTask` and `OfflineMapTask` now support exporting tiles from an Image Service to support offline usage.
- **Time-based expiration** of mobile map packages and mobile scene packages is now supported in ArcGIS Runtime. An author can choose either a hard expiration where the package is not usable after the expiry date, or a soft expiration where a warning message my be displayed to the user but the package is still usable after the expiry date. Packages with such expiration constraints can be created using an upcoming version of ArcGIS Pro.

Geocoding

The **next generation locator format** (`.loc` file with associated `.loz` data) available in ArcGIS Pro 2.3 is now supported in runtime. It offers better result matching, faster offline performance and a more compact file storage. All capabilities available with the previous format such as finding addresses, locations, and suggestions are supported with the new format using the same `LocatorTask` API.

Coordinate systems, projections, and transforms

- Coordinate systems—The projected coordinate systems for Las Vegas, Pima County, Serbia, and Saudi Arabia were added.
- Projections—The Equal Earth and Peirce Quincuncial projections were added.
- Transformations:
  - Transformations for Slovenia, Saudi Arabia, Georgia (country), St. Pierre and Miquelon, Switzerland, and 19 others were added.
  - File-based geographic/datum transformations were added for Belgium and Switzerland. These files are part of the projection engine .zip file that can be downloaded at https://developers.arcgis.com/downloads/data (requires sign in).

Memory optimizations

A number of enhancements have been made to conserve the amount of application memory required to display 3D scene layers. These enhancements include:

- supporting compressed ETC2 textures for mobile devices. ETC2 textures can be authored with ArcGIS Pro.
- releasing texture memory more aggressively
- using pre-defined or on-the-fly oriented bounding boxes (to improve culling of data not required to be rendered)
- optimizing data structures for vertices and indexing them to reduce redundancy
- and many more.

These enhancements have led to **a memory reduction of 40-50%** in many cases.

API changes

Changes to feature selection API

These methods that were deprecated at 100.4.0 no longer function at 100.5.0:

- FeatureLayer.selectionWidth

Copyright © 1995-2020 Esri. All rights reserved.

Behavior changes

- **Selection** of features and graphics has been completely redesigned and overhauled in order to improve performance by taking advantage of the GPU and also to accommodate new functionality in this release such as annotation and reference scale. As a result, the appearance of selected features and graphics has changed from previous releases. Visually, selection is now more bold, and displayed on top of all content in a map view or scene view rather than on just the selected features or graphics, making it more distinguishable and easier to spot in bright ambient light conditions. Read more about this change in this blog post.
- Starting from version 3.0 of mobile map packages, maps are returned from the package **in the order in which they were added to the package**, rather than alphabetically.
- Default parameters for taking a map offline using on-demand and pre-planned workflows are now based on the options chosen by the map author in the **offline settings** of the web map.
- Surface now **returns elevation from the top most elevation source** (i.e the source that was added last) when two or more overlapping sources contain information for the same location. In previous releases the bottom most source was used. This change now makes it consistent with how elevation data is displayed in a sceneview for overlapping sources, and is also consistent with how content is returned from operational layers (for example, in identify operations)
- When opening web scenes, scene layers containing integrated mesh data in the I3S format are now returned as `IntegratedMeshLayer` type instead of the generic `SceneLayer` type that was previously used.
- Visibility analysis such as Viewshed and LineOfSight are now **computed based on the geometry** of the observer/target. Previously, the analysis was computed based on the center of the symbol used by the observer/target
- `SketchEditor.EditConfiguration` returns `null` until the sketch editor is started (when `SketchEditor.StartAsync` is called, in other words). In previous versions, it was possible to make changes to the edit configuration that would then be overridden when `StartAsync` was called.

Deprecations

Support for iOS 10 was deprecated in a previous release and ArcGIS Runtime SDK 100.4.0 was the last release to support it. A minimum of iOS 11 is required at 100.5.0.

Support for development of UWP apps for the ARM32 architecture was deprecated in a previous release and ArcGIS Runtime SDK 100.5.0 is the last release to support it.

Issues resolved

- BUG-000118198 Highest LOD available in TPK not rendered at the appropriate map scale
- BUG-000116810 QueryFeatures performance is slower in v100.3 Runtime when compared to v10.2.x.
- BUG-000115663 Setting MaxFeatures parameter when performing a spatial query returns inconsistent results
- BUG-000108264 RelationshipInfo is not returned properly from a mobile map package.
- BUG-000117456 Popup.GetformattedValue() method crashes the application when the Popup field populates with minimum Date Value (1/1/0001).
- BUG-000092734 Some of the KML files do not get loaded in 3D Scene view
- BUG-000116502 When quickly generating animated simple marker symbols, the style of the feature layer in the map is replaced by the style of the animated simple marker symbol.

Copyright © 1995-2020 Esri. All rights reserved.

- ENH-000115061 Request for disabling automatic realignment of the Mapview when the Mapview.LocationDisplay.IsEnabled is set to False.
- BUG-000116792 Layers in a mobile map package created using ArcGIS Pro do not respect label stacking properties.
- BUG-000108637 The ReverseGeocodeAsync method does not return all of the ResultAttributeNames found in the GeocodeParameters
- BUG-000104032 Collada models are not rendered correctly with ModelMarkerSymbol.
- BUG-000103661 ArcGIS Runtime doesn't honor Arcade expression while consuming as a web map.
- ENH-000116571 Include information in the Local Server documentation that to create geoprocessing packages in ArcGIS Pro that are ArcGIS Runtime enabled, one must use the Package Result tool
- BUG-000115806 Rendering geodatabase crashes intermittently in ArcGIS Runtime application while the geodatabase is displayed without any issue in ArcGIS Pro.
- Invalid argument error occurs when adding a new or existing feature with geometry to the table
- BUG-000114231 A feature service layer with a picture marker symbol with x-offset or y-offset set is not displayed properly as a feature layer
- BUG-000117217 Graphics floating in draped mode in a scene (dynamic mode only)
- BUG-000116867 Deadlock in SceneView on destruction
- BUG-000114143 Military symbols fail to display correct when resizing them in a Unique Value Renderer.
- BUG-000117464 Crash due to graphics list being modified at the same time the graphic objects are deleted
- BUG-000117464 Crash due to graphics list being modified at the same time the graphic objects are deleted
- BUG-000120016 Adding a multipoint object containing more than 250 points as a graphic using ArcGIS Runtime crashes the MapView.
- BUG-000116279 Cannot create TileCache from a path on Linux only
- BUG-000116794 Loading MapImage Sublayers causes MapImageLayer to draw incorrectly
- BUG-000117070 When creating features in a downloaded offline map, Runtime returns the error message, "The table is not editable" when the feature service dataset uses the database time for editor tracking.
- ENH-000116951 How to handle Integrated Windows Authentication in ArcGIS Runitme?
- BUG-000118069 System.Threading.Tasks.TaskCanceledException thrown on assigning the MapView to System.Windows.Forms.Integration.ElementHost in ArcGIS Runtime SDK WPF app.
- PKI Authentication Crash. See this GeoNet post
- BUG-000117627 Using SketchEditor, an arrow is drawn in the reflected direction when the MapView displays the whole world map.
- BUG-000118762 ServiceFeatureTable Query operation with complex geometry results in exception "Invalid URI: URI string too long"
- BUG-000117733 The 'GeometryChangedEventArgs' NewGeometry property returns an incorrect number of points after a point is deleted using 'MapView.SketchEditor.DeleteCommand.Execute'.
- BUG-000116526 ArcGIS Runtime must use POST request instead of GET when making FeatureTable.QueryFeaturesAsync are too long.

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000118943 Slow performance of drawing tiles when zooming quickly using the mouse wheel.
- GeometryUtils.ValidateSpatialReference : Provide helpful error message when SR do not match. See this GeoNet post.
- BUG-000120306 ArcGIS Runtime for .Net shows blank Map when it falls back to Software Rendering
- SketchEditor.EditConfiguration should be null when there is no active draw/edit. See this GeoNet post.
- PortalItem: GetDataAsync for secured portal item for download with OAuth header fails with Bad Request. See this GeoNet post.
- ArcGISWebException: null Message, does not fallback to use Detail. See this GeoNet post.
- SketchEditor: Adding vertex with magnifier when there's a pan. See this GeoNet post.
- SketchEditor: Freehand draw should not respond to flick. See this GeoNet post.
- Forms TabbedPage Android: switching between tabs breaks mapview. See this GeoNet post.
- SketchEditor: Cannot undo moved vertex/geometry. See this GeoNet post.
- Forcing WPF to render in software causes the map to not render. See this GeoNet post.
- WMTSLayer may not use the right ResourceURLs, causing tiles to fail loading. See this GeoNet post.
- Cannot load WMTS services with empty Keywords. See this GeoNet post.
- WMTSLayer: URL template variables should be replaced in a case-insensitive way. See this GeoNet post.

Known issues

- The *.loz locator format is not supported by 32-bit apps. You may get an EXC_BAD_ACCESS error or an access violation when calling SuggestAsync on a composite locator from a mobile map package.
- BUG-000119044 Shapefile polygon created from the Cut Polygons Tool in ArcMap causes ArcGISRuntimeException when updating the shapefile from an ArcGIS Runtime application.
- BUG-000117942 The deleteFeature() method fails to delete a feature completely from a shapefile..
- BUG-000119455 ArcGIS Runtime's GeocodeResult.Label property returns no value for some geocoded addresses.
- BUG-000118590 A mobile map package (.mmpk file) created using decorations in the cartographic line symbol does not return the expected results in an ArcGIS Runtime app.
- BUG-000120409 WMTS service does not load if the tile matrices of the service in the GetCapabilities file are not in sequential order.
- BUG-000120079 Labels do not honor frequency settings when a Vector Tile Package (.vtpk) is deployed in ArcGIS Runtime.
- BUG-000118445 MapView fails to zoom out to the full extent of larger basemap when a basemap with smaller extent is replaced by the basemap with the larger extent.
- BUG-000115930 Heavy scale dependent feature layer affects the performance of other feature layers in an .mmpk file.
- BUG-000107742 Analysis Services cannot be used with an AGSGeoprocessingTask in an ArcGIS Runtime app.
- BUG-000118443 The KMZ file does not display image as a photo overlay in ArcGIS Runtime.
- BUG-000119797 ArcGIS Runtime does not honor custom ArcGIS Pro Billboard3D stylx symbology.
- BUG-000119575 The AngleAlignment Property of a MultilayerPointSymbol is not honored in a 3D scene but honored in SimpleMarkerSymbol.

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000118574 3D scene in an ArcGIS Runtime app (WPF) does not render properly when using DTED files as the elevation source.
- BUG-000118517 An ArcGIS Runtime SDK for .NET app fails to render 3D data layer in a web scene from ArcGIS Online.
- BUG-000119384 An ArcGIS Runtime SDK 100.4 app displays a blurry image of an image service hosted in System.Windows.Forms.Integration.ElementHost.
- `Esri.ArcGISRuntime.ArcGISRuntimeException`: DirectX failure CreateQuery code = 0x887a0005. See this GeoNet post.
- Because some OGC services, such as WFS, WMS, WMTS, and KML, do not challenge clients to log in (with an HTTP/401 response code), the initial request for secured services must include login information. The following workaround is required to implement this workflow since .NET's HTTP client will not send a `NetworkCredential` unless challenged for one.
  - **Workaround**: Handle the `HttpRequestBegin` event to add network credential information to an authorization header.

```
private void EagerlyAddNetworkCredentials(object sender, HttpRequestMessage e)
{
    var anyCred = AuthenticationManager.Current.FindCredential(e.RequestUri,
AuthenticationType.NetworkCredential);
    if (anyCred is ArcGISNetworkCredential runtimeNetCred)
    {
        // If a NetworkCredential is found for this request, but we are still waiting to be
challenged...
        if (!e.Headers.Contains("Authorization"))
        {
            // Generate an HTTP Basic Authorization header per RFC 2617
            var netCred = runtimeNetCred.Credentials.GetCredential(e.RequestUri, "Basic");
            var rawCredentialBytes =
Encoding.GetEncoding("ISO-8859-1").GetBytes($"{netCred.UserName}:{netCred.Password}");

            // Add it to the request
            e.Headers.Add("Authorization", $"Basic
{Convert.ToBase64String(rawCredentialBytes)}");
        }
    }
}

// Handle HttpRequestBegin by adding login info to an authorization header for the request
ArcGISHttpClientHandler.HttpRequestBegin += EagerlyAddNetworkCredentials;

// Add valid network credentials to the AuthenticationManger and load the service ...
```

- Http authentication always returns "401 (Unauthorized)" for Xamarin Android and Xamarin iOS when using the native `HttpClient` implementation (`AndroidClientHandler` and `NSUrlSessionHandler` respectively). The code shown above for pre-authorizing in the `HttpRequestBegin` event can be used to workaround this issue.
- Error when authenticating with a portal using IWA: "Your login session has been reset. Please try again.". This error may occur when the credentials used to authenticate have been updated (to a different user) but the cookie from the previous login is still stored.
  - **Workaround**:Explicitly expire the cookies for previous logins.

```
var credList = AuthenticationManager.Current.Credentials.ToList();
var clientHandler = new ArcGISHttpClientHandler();
foreach (var credential in credList)
{
    if (credential.ServiceUri != null)
    {
```

Copyright © 1995-2020 Esri. All rights reserved.

```
clientHandler.CookieContainer.GetCookies(credential.ServiceUri).Cast<Cookie>()?.ToList()?.ForEach(c
=> c.Expired = true);
    }
    AuthenticationManager.Current.RemoveCredential(credential);
}
```

Related topics

Release notes
Release notes for 100.7.0
Release notes for 100.6.0
Release notes for 100.4.0
Release notes for 100.3.0
Release notes for 100.2.0
Release notes for 100.1.0
Release notes for 100.0.0

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.4.0

This page provides details about enhancements in the 100.4.0 release of ArcGIS Runtime SDK for .NET. It also lists this release's deprecations, resolved issues and known issues.

For release notes for previous releases, see Related topics at the end of this topic.

Enhancements

This section describes new features and improvements.

Offline maps

- You can now gain fine-grained control over how individual layers are taken offline. Use the layer specific overrides to adjust settings such as whether to omit layers, the area of interest, attribute queries and attachments behavior on a per-service basis.
- Support has been added for a "fail fast" mode when taking maps offline. By default, a job will run to completion even if a layer or table has problems.
- The offline local item has been enhanced to allow you to update meta-data on the client.
- Additional meta-data has been added to Job messages. This information includes the time-stamp and severity of a given message.

Behavior changes include:

- The on-demand workflow will now honor the scale range of a tiled layer as specified by the web map. This behavior helps to ensure that only the tile data required by the offline map is downloaded.
- The default behavior for taking a map with related tables offline has been updated such that only the related records from a destination table will be included. All unrelated records will be excluded. This ensures that the offline map contains only the related records that are relevant to the offline features. If you wish to revert to the old behavior (taking all records offline) you can adjust the value of destination table row filter provided by the generate offline map parameters.

Support for MrSID raster format

Multiresolution Seamless Image Database (MrSID) raster format is based on a proprietary compression technique for maintaining the quality of large images. It allows for a high compression ratio and fast access to large amounts of data at any scale without having to decompress the entire file. You can display MrSID data using raster layers in both 2D maps and 3D scenes, and also apply raster functions. Generations 2, 3, and 4 of MrSID are supported.

KML

Runtime now supports reading and displaying KML content in both 2D maps and 3D scenes. You can use KML and KMZ files either from a location on disk or on the web. At this release, the following features are supported in KML version 2.2:

- Placemarks (point, line, and polygon), ground overlays, labels, and 3D models
- Screen overlays, Photo overlays
- Network links, including support for automatic refresh intervals
- Time awareness - KML layers participate in time filtering
- Altitude mode (Absolute and Clamp-To-Ground/draped)

Copyright © 1995-2020 Esri. All rights reserved.

- Ability to identify Placemarks and get Balloon popup content

API changes

UI controls namespace change for Xamarin.Android

In previous releases, Esri UI controls (including `MapView` and `SceneView`) could be accessed through the `Esri.ArcGISRuntime.UI` namespace in Android layout files (.axml). In 100.4, the control namespace is `Esri.ArcGISRuntime.UI.Controls`. In your .axml Android layout files, change all instances of `Esri.ArcGISRuntime.UI` with `Esri.ArcGISRuntime.UI.Controls` to avoid run time errors when loading layouts.

Deprecations

Support for development of UWP apps for the ARM32 architecture was deprecated in the previous release and ArcGIS Runtime SDK 100.4.0 is the last release to support it.

Support for iOS 10 was deprecated in the previous release and ArcGIS Runtime SDK 100.4.0 is the last release to support it. A minimum of iOS 11 will be required at 100.5.0.

Issues resolved

- BUG-000115436 Gap at horizon when edge of scene is below sea level and elevation exaggeration is greater than 1.
- BUG-000116204 There are excessive "Fork" instructions in Driving Directions when passing through intersections with turning slips
- BUG-000116629 Route Task resource string ArgumentNullException: 'Value cannot be null. Parameter name: format'
- BUG-000112142 Application crashes after repeatedly changing the value for the display setting in Windows 10: Scale and Layout
- BUG-000113103 The heading direction does not update correctly when device orientation is changed in CompassNavigation mode for the LocationDisplay in Xamain Forms application.
- BUG-000114899 SketchEditor doesn't draw the graphics but pans the map right after changing the zoom level.
- BUG-000116794 FeatureLayer.LoadAsync causes layers to draw incorrectly
- Screen coordinates on Xamarin Forms Android are now measured in density-independent pixels (DIPs) rather than physical pixels. This is now consistent with the other Xamarin Forms platforms.

Known issues

- BUG-000115332 Invalid argument error occurs when adding a new or existing feature with geometry to the table
- BUG-000114932 Labels do not follow through as the map view is zoomed or panned, unlike in ArcGIS Pro.
- BUG-000115663 Setting MaxFeatures parameter when performing a spatial query returns inconsistent results
- BUG-000115316 Some layers in the Mobile Map Package don't render in ArcGIS Runtime application.
- BUG-000116810 QueryFeatures performance is slower in v100.3 when compared to v10.2.x.

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000116792 Layers in Mobile Map Package created using ArcGIS Pro do not respect label stacking properties.
- BUG-000114058 Zooming in on an area that crosses the International Date Line results in the entire basemap.
- BUG-000113832 DictionaryRenderer is not rendering certain oceanographic point symbols from MIL-STD-2525D correctly.
- BUG-000109405 Labels generated with python in an Mobile Map Package do not display
- BUG-000108786 Changing GraphicsRenderingMode from Dynamic to Static and vice versa changes rendering direction of composite symbol placement effect

  Note:
  The supported workflow is to use a Style file or any other sort of transfer mechanism where the CIM version is attached, or you have to upgrade your CIM to the latest Pro version for Runtime.

- [UWP only] PortalItem: GetDataAsync for secured portal item for download with OAuth header fails with Bad Request. See Geonet post
- Custom ServiceImageTiledLayer zoom visibility. See Geonet post
- BUG-000116502 When quickly generating animated simple marker symbols, the style of the feature layer in the map is replaced by the style of the animated simple marker symbol.
- BUG-000116526 Runtime must use POST request instead of GET when making FeatureTable.QueryFeaturesAsync requests that are too long.
- BUG-000116547 Esri.ArcGISRuntime.Http.ArcGISWebException - Failed to create service - exception thrown by ArcGIS Runtime local server 100.3.0 when trying to run a local service from a local geoprocessing package
- BUG-000114871 Navigation bars get larger after changing the screen orientation in Xamarin Forms (iOS
- BUG-000111563 Japanese field names appear garbled when shapefiles with the Shift-JIS character code are loaded in a Xamarin Android or Xamarin iOS app.

Related topics

Release notes
Release notes for 100.7.0
Release notes for 100.6.0
Release notes for 100.5.0
Release notes for 100.3.0
Release notes for 100.2.0
Release notes for 100.1.0
Release notes for 100.0.0

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.3.0

Version 100.3.0 of ArcGIS Runtime (also known as Update 3) is a significant update, with a focus on performance improvements, greater parity with the full ArcGIS platform, and support for 3D Web GIS. This page provides details about these improvements and describes additional enhancements. It also lists deprecations, resolved issues, and known issues.

For release notes for the previous release, see Related topics near the bottom of this page.

Enhancements

This section describes new features and improvements.

Support for .NET Standard

This release adds support for building components that reference ArcGIS Runtime functionality and target .NET Standard 2.0 for use with your UWP, WPF, Xamarin.Android, Xamarin.iOS, and Xamarin.Forms (Android, iOS, and/or UWP) applications. To build and use a component that targets .NET Standard 2.0:

- Add a reference to the Esri.ArcGISRuntime NuGet package to your component project.
- From your application project(s), add a reference to the Esri.ArcGISRuntime NuGet package. This will ensure the required ArcGIS Runtime components are deployed for your applications.

The introduction of support for .NET Standard 2.0 in Esri.ArcGISRuntime required updates to the minimum requirements:

- UWP:
  - Microsoft.NETCore.UniversalWindowsPlatform 6.0.4 or later
  - Windows 10 Version 1709 (OS Build 16299) (Fall Creators Update) or later
  - Visual Studio 2017 15.4 or later
- WPF: Microsoft .NET 4.6.1 or later
- Xamarin.Android: 8.0 or later (latest Stable channel recommended)
- Xamarin.iOS: 10.14 or later (latest Stable channel recommended)
- Xamarin.Forms: 3.0 or later and Microsoft .NET Framework 4.7.1 or later (latest recommended)

Preparing to update:

- We recommend upgrading your projects from project.JSON to PackageReference.
- WPF: Ensure your project targets Microsoft .NET Framework 4.6.1 or later.
- Xamarin Forms: Update your project to reference Xamarin.Forms 3.0 or later. **Note**: Xamarin.Forms version 3.0 requires Visual Studio 2017 15.4 minimum.
- UWP:
  - Set your project Target version and Min version to 16299 or later.
  - Update your project reference for Microsoft.NETCore.UniversalWindowsPlatform is 6.0.1 or later.

Web scenes

As part of the improved support for 3D Web GIS capabilities of the ArcGIS Platform, you can now display web scenes from ArcGIS Online or ArcGIS Enterprise in your ArcGIS Runtime based apps. Web scenes can be quickly authored by apps such as ArcGIS Pro and Scene Viewer and then easily consumed by your apps, respecting the content, symbology, pop-ups and other

Copyright © 1995-2020 Esri. All rights reserved.

settings that the author configured in the web scene. For more information about Runtime support for web scenes, see Runtime support for web scenes.

In addition, 3D content originating from scene layers can now be identified and selected, and their display can be customized through renderers in the web scene. And feature layers can be displayed with 3D symbols originating from web styles in ArcGIS Online or ArcGIS Enterprise.

Scene analysis

- Interactive distance measuring is now supported in 3D. You can measure horizontal, vertical, and direct distances between two locations in a scene view.
- Viewshed analysis has been enhanced to support full 360 degrees horizontal field of view. Previously, only a maximum of 120 degrees was supported.

Note:
In previous versions, offsets for scene analysis, such as viewshed and line of sight, were incorrectly applied from the center of the model symbol. Rather, the scene analysis offsets should have been applied from the geometry location. This has now been resolved and there may be some cases where offsets used by your analysis may need to be changed or removed.

Scene view memory management

Scene view has been enhanced to keep memory utilization under a threshold when working with large amounts of 3D content on Android devices to prevent the application from running out of memory. The default limit is calculated on the basis of total system memory available and is shared by all scene views in the application. You can choose to override the limit through the `MemoryLimit` setting on the scene view. Refer to the API reference for more details.

Arcade

Arcade expressions are now supported:

- In pop-ups for layer types (for example, feature layers) that support pop-ups. They're defined using the expression property on a pop-up definition.
- In label definitions for feature layers and graphic overlays. They're defined in JSON using the expression property on the labelExpressionInfo object.
- In visual variables within renderers defined for feature layers in web maps and mobile map packages. Feature layers must be rendered in static mode to honor visual variables in renderers.

See this SDK's API reference for details on using Arcade. For general information on Arcade and Arcade's function reference, see the ArcGIS Arcade guide.

WMS

WMS support has been enhanced:

- WMS layers are now time aware and support filtering their contents based on a time range or time instant.
- You can discover available styles in the service and set the style of a layer to customize its display.
- You can now specify custom parameters to be sent to the service. This is useful, for instance, if the service requires custom authorization parameters in the HTTP requests.

Copyright © 1995-2020 Esri. All rights reserved.

- WMS layers now automatically display attribution information of the service in a map view or scene view .

Labeling

Improvements have been made to provide finer control over label placement. These refinements allow for a more consistent display of labels when using mobile map packages (`.mmpk` files) generated from ArcGIS Pro. You can now do the following:

- Allow overlap of a label with other features or labels based on weights to assign relative importance and define priorities when there is a conflict
- Specify an offset distance of a label from the symbol based on screen units.
- Stack multiple lines of text with custom line separators and choose whether line break should occur before or after the separator, whether the separator should be visible, etc. Previously, only whitespace characters were considered for inserting line breaks.

Query support for map image layer

You can now query sublayers and tables from a map image layer in the same way you query feature layers. The query can be based on attributes and/or spatial relationships and can also be used to return summary statistics or related features. Also, a new method has been added to conveniently load all the sublayers and tables contained in the layer so that their data can be accessed.

Military symbology

Support for App-6(B) and App-6(D) military symbology specifications has been added. You can use the corresponding style files to display symbols based on these standards in your ArcGIS Runtime apps.

Guide documentation

- Expanded and reorganized layers topics. A new table summarizes supported layers and their capabilities.
- Expanded and reorganized map and scene documentation.

Breaking changes

.NET Standard defines System.Drawing.Color as the primary color API. ArcGIS Runtime SDK for .NET now follows this pattern and uses System.Drawing.Color in place of Windows.UI.Colors on UWP and System.Windows.Media on WPF.

Deprecations

Support for development of UWP apps for the ARM32 architecture is deprecated.

Support within Visual Studio 2015 for project templates (WPF, Xamarin.Android, and Xamarin.iOS) and local NuGet repository is deprecated. NuGet packages can be referenced within Visual Studio 2015 from NuGet.org or downloaded to the existing `Esri` local NuGet repository (`%localappdata%\Esri\NuGet`). A minimum of Visual Studio 2017 will be required by the ArcGIS Runtime SDK for .NET Visual Studio extension (.vsix) at a future release.

Support for iOS 10 is deprecated. A minimum of iOS 11 will be required at a future release.

Copyright © 1995-2020 Esri. All rights reserved.

Issues resolved

- BUG-000112740 The first feature added to an empty shapefile does not display until a second feature is drawn.
- BUG-000114262 An error may occur in EncExchangeSet::getPaths resulting in inability to load EncLayer
- BUG-000110150 Application with a universal transverse Mercator (UTM) spatial reference may crash when the MapView is centered on a location outside the full extent of all layers
- BUG-000109558 Opening a mobile map package fails with the error, "Internal Error" when the map in the mobile map package contains German umlauts in the map name.
- BUG-000114330 When WMTS services have a reverse order for tile matrices, the layer requests incorrect tiles from the service resulting in 404-Not Found error and layer not added to map.
- ENH-000107768 Request to add Runtime API to load a Web Scene PortalItem to Scene
- BUG-000111237 Application may crash when zooming into a viewpoint with very small geometry.
- BUG-000109728 OfflineMapSyncTask doesn't update the feature service in the webmap if the spatial reference of the feature service isn't Web Mercator.
- BUG-000111867 CompositeSymbol doesn't display properly when it is applied to a Graphic with a Geometry of Envelope
- BUG-000112435 Vector tiled layer constructed from a vector tile package (.vtpk) containing very large sprites crashes the Runtime application on the machine with a high screen display resolution
- BUG-000112114 "Out of Range" error is received while calculating a route when multi-part polylines are present in the source feature classes of the network dataset
- ENH-000108221 Request to add MapServer/DynamicLayer support for FeatureLayer or ServiceFeatureTable
- BUG-000106101 Vector tiles may fail to display some Chinese characters.
- BUG-000109215 Symbology layer effects in Mobile Map Package are not rendered correctly.
- BUG-000114411 Elevation exaggeration of Surface does not give expected results when the image service has a negative elevation
- BUG-000112713 Faulty display of military symbols in 3D
- BUG-000111565 Lines created on the GraphicsOverlay with "Dynamic" Rendering mode on 3D doesn't render completely when zooming in
- BUG-000113632 Feature collection points continue to draw although they do not satisfy the definition expression.
- BUG-000111110 Disconnecting a Windows 10 laptop from an external monitor set as the primary display while a Runtime app is running causing the app to crash or freeze.
- BUG-000113591 GraphicCollection.Add causes a tabbed-control Runtime .NET app to crash upon clicking the tab
- BUG-000111214 Application crashes on startup and after changing the screen resolution of the monitor's display.
- BUG-000113103 The heading direction does not update correctly when device orientation is changed in CompassNavigation mode for the LocationDisplay in Xamain Forms application.
- BUG-000112443 Unhandled exception when NavigationCacheMode is Enabled
- BUG-000109636 Application with multiple MapViews crashes at launch intermittently
- BUG-000114015 WPF application crashes when them is changed to a High Contrast theme

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000106702 Point screenPoint returns (0,0) (on physical devices only) when using the IdentifyGraphicsOverlaysAsync() method
- ENH-000109715 Add Legend UI to the ArcGIS Runtime SDK Toolkit
- BUG-000110752 Unable to download .geodatabases from some FeatureServices when using the OfflineMap workflow.
- BUG-000112461 Labels are missing if features co-exist when viewing an offline geodatabase in ArcGIS Runtime application.
- Unhandled exception Microsoft C++ exception: Esri_runtimecore::Runtimecore_net::Dx_exception occurred. See GeoNet discussion
- KERNELBASE.dll exception occasionally thrown since upgrading to v100 from v10. See GeoNet discussion
- FeatureCollectionTable throws OBAC exception when geometry is updated. See StackExchange discussion
- Runtime crashes with JNI exception on Android devices. SeeGeoNet discussion
- Change MapView Parent Quickly Cause Map Blank. See GeoNet discussion

Known issues

- BUG-000099242 Japanese labels in a mobile map package are not displayed
- BUG-000113630 Application may crash when loading VectorTileLayers that require large amount of memory
- BUG-000101144 Rotating a vector tile map causes the screen to flash.
- BUG-000110656 Gradient fill symbols are displayed as black BUG-000113946 Performance issue of MapView.IdentifyLayerAsync method
- BUG-000109949 Locator Task from a Mobile Map Package may return suggestions even for invalid addresses.
- BUG-000109987 Labels generated by python in a map package (.mpk) may not display correctly
- BUG-000109554 Aplication may encounter an error when trying to display a tile package (.tpk) containing only one Level of Detail.
- BUG-000108845 Mosaic Dataset doesn't display NoData pixels properly
- BUG-000111375 Locator created from ArcMap 10.5 does not provide 'Match_addr' or 'Label' even if the score is 100.
- BUG-000107292 Setting the opacity of a FeatureLayer displayed in a SceneView causes the layer to turn white and opaque.
- BUG-000104032 Collada models are not rendered correctly with ModelMarkerSymbol.
- BUG-000107500 Layer in a Mobile map package based on a unique value renderer may not render correctly
- BUG-000103301 Some military line symbols do not display in the correct location
- BUG-000111519 Without elevation source the SceneView::screenToLocation function provides invalid "z" values
- BUG-000113128 UniqueValueRenderer fails to draw features when using values of type unsigned int
- BUG-000114143 Military symbols fail to display correctly when resizing the symbols in a unique value renderer
- BUG-000114174 Some mil2525c symbols fail to render properly

- BUG-000109943 When creating the points, if the coordinate value is lower than decimal point third place,the raster is added to the map with unexpected rendering result.

Copyright © 1995-2020 Esri. All rights reserved.

- BUG-000109847 RouteTask.SolveRoute fails with ArcGISRuntimeException "Invalid directions language" with certain language string set as RouteParameters.DirestioncsLanguage
- BUG-000109177 Xamarin.iOS Storyboard Error: "Custom Components are not being rendered because problems were detected"
- BUG-000110854 Xamarin.iOS and Xamarin.Forms.iOS application crashes when cancelling OAuth login.
- BUG-000106812 Editing vertices is successful even when wrap-around is disabled
- BUG-000112142 Application crashes after repeatedly changing the value for the display setting in Windows 10: Scale and Layout

Related topics

Release notes
Release notes for 100.7.0
Release notes for 100.6.0
Release notes for 100.5.0
Release notes for 100.4.0
Release notes for 100.2.0
Release notes for 100.1.0
Release notes for 100.0.0

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.2.0

Version 100.2.0 of ArcGIS Runtime SDK for .NET (iOS), also known as Update 2, is the second major update to 100.0.0. This topic describes what's new, provides a list of known limitations, and more.

For release notes for previous releases, see Related topics at the bottom of this page or sign in to access the Downloads page and download the guide PDF for the version you want.

Enhancements

Sketch Editor

- Methods on `SketchEditor` class support programmatic manipulation of vertices relative to the selected vertex (insert after, move selected, and remove selected).
- `SketchVertex` class adds the ability to get/set the selected vertex and receive notifications when the selected vertex changes.
- `RequireSelectionBeforeDrag` property provides an alternative editing mode to help prevent accidental editing of point features.
- Sketch creation modes that rely on tap (point, polyline, and polygon) now support adding a vertex when using tap and hold to display the magnifier.

Popups

- New `PopupManager` and `PopupAttachmentManager` classes provide enable you to more easily create the UI experience for viewing and manipulating popups.

SceneView

- The SceneView now continues to rotate following a flick interaction. This behavior can be enabled and disabled via the `Esri.ArcGISRuntime.UI.SceneViewInteractionOptions.IsFlickEnabled` property.

New layers

A new WMS layer is available that can display content from **OGC-WMS** services in maps and scenes. You can identify features that are displayed, and generate a legend for them. Only WMS version 1.3 is supported at this release, but support for other versions will be added in a future release. For more information, see the WMS layer section in "Layers and tables."

A new ENC layer is available that can display content from **ENC** (electronic navigational charts) data in S-57 format. ENC is a vector chart that conforms to the IHO specifications contained in Publication S-57, which is a transfer standard for vector data. The ArcGIS Runtime implementation follows the S-52 Presentation Library 4.0 specification for rendering. You can identify features that are displayed, select features, and change various display settings for view groups, text, and other elements such as isolated dangers, contours, color scheme, and so on. S-63 is not supported at time of publishing.

New data formats

Direct read of **Shapefile** datasets is now supported. Shapefiles can be added as a feature layer for display in maps and scenes. You can also add and edit features in the dataset through the shapefile feature table.

Copyright © 1995-2020 Esri. All rights reserved.

Support for the **OGC GeoPackage** format has also been added this release. You can add vector and raster datasets in a GeoPackage to your maps and scenes as feature layers and raster layers respectively. You can also add and edit features in an existing GeoPackage feature table.

Rendering improvements

Feature layers can now be rendered **dynamically** in addition to statically, just like graphics overlays. You can set the rendering mode at the map or scene level via load settings or on a layer by layer basis at the feature layer level. Dynamic rendering improves the appearance and interactivity of features during map or scene navigation. Feature layers containing point geometries are rendered dynamically now by default and their symbols remain screen-aligned in map views and will be billboarded in scene views. Feature layers containing polygon or polyline geometries are still rendered statically by default, but you can choose to render them dynamically to allow for 3D behavior such as extrusion based on feature layer attributes and surface placement based on z values

Display performance of graphics overlays has also been improved when updating large volumes of graphics. In some cases, the performance as a measure of frame rate is 2x faster compared to previous releases.

New **multi-layer symbol** types have been introduced to better represent working with feature layers that contain advanced cartography. At this release, these symbol types cannot be created by developers but can be authored in ArcGIS Pro and deployed through feature services, Mobile Map Packages, and Mobile Style files for use in the runtime. In future releases, these symbol types will be further expanded so that developers can produce sophisticated cartographic effects by combining multiple symbol layers.

Time support

You can now apply a time extent to map views and scene views to filter the display of content from time-aware layers. Time-aware layers include feature layers, map image layers, and raster layers. You can apply time offsets to time-aware layers, which can be used to compare data over time. You can also specify temporal parameters when querying a feature table.

Analysis

With the new Scene Analysis API you can define a variety of analyses to be performed using data displayed in the current 3D scene view, then render results that are updated dynamically. This release includes two types of scene visibility analyses: **viewshed** and **line of sight**. Viewshed highlights areas in your 3D scene that are visible from a given observer. Line of sight shows which segments are visible along a line drawn between an observer and a target location. For either type of analysis, the observer and/or target may be moving or stationary.

The new **statistics query** API allows you to get any of the following statistics for a specified field in a feature table: Sum, Average, Count, Minimum, Maximum, Standard Deviation, or Variance. In the statistics query parameters, you can define filters for features to include in the statistics (based on attributes, spatial relationships, or time extent) as well as how the results are grouped and sorted.

Coordinate systems and transformations

Geographic transformations (or datum transformations) can now be discovered, defined, used in `project` function of the geometry engine class, and chosen to be used by default. Prior to this release, the most suitable transformations have been used automatically whenever data is projected. A new transformation catalog class lets you look up a list of the best transformations to use when projecting between two spatial references that have different datums. The new method, `getTransformationsBySuitability`, provides a list of applicable transforms for the two spatial references you provide,

Copyright © 1995-2020 Esri. All rights reserved.

ordered by suitability. You can even pass in a specific envelope to get back transforms suitable for that specific area. You can define a transformation by well-known ID (WKID) or create a custom transformation using well-known text strings. You can also change the default transformation that is used internally via `gtdefaults_overrides.json`. Both equation-based and grid-based transformations are supported. You can use `isProjectionEngineDataMissing` to determine whether required grid transformation data files are missing. The grid transformation files needed by your app can be downloaded from the downloads page. Read more about geographic transformations in the topic Spatial references.

Offline maps—preplanned workflow

This release enhances on-demand workflows with support for exporting and downloading vector tile packages from vector tile map services hosted by ArcGIS Online or ArcGIS Enterprise. Vector tile packages contain a default style to define how tiles are rendered. Vector tile layers can also reference custom styles as resources in a portal item. This release also supports downloading and applying custom style resources to vector tile layers on the client.

This release expands your ability to take maps offline with the new preplanned workflow. Offline maps allow your users to continue being productive even when their network connectivity is poor or non-existent. The preplanned workflow supplements the existing on-demand workflow by providing an alternative approach of allowing the map author to define and pre-create offline map areas instead of the field worker. Your field workers can then download map areas as required. With both of these offline workflows, the field worker can synchronize any updates to operational data when connectivity is restored.

Transactional editing

Transactional editing is now supported in the geodatabase and geodatabase feature tables. This allows you to perform a number of edits and then choose to commit all of them together as one unit, or roll them all back if any of them encounters an error. Nested transactions are not currently supported.

Localization

Directions returned by route task and closest facility task are available in 10 additional languages - Danish, Finnish, Hindi, Croatian, Indonesian, Norwegian Bokmål, Romanian, Serbian, Vietnamese, and Chinese (Taiwan). If the requested language isn't available, directions fall back to a default language instead of failing. Furthermore, error messages returned by Route, Service Area, and Closest Facility tasks are now consistent and translated into all supported languages. As part of these changes, some `DirectionMessage` types (Length, Time, Summary, TimeWindow, ViolationTime, ServiceTime, EstimatedArrivalTime, CumulativeLength) are no longer reported. These messages did not respect platform locale settings for formatting numbers, dates, and times messages, and the information they contained is available through other properties in the results.

Deprecations

Version 100.2 is the last release to support:

- .NET Framework versions 4.5.2 and 4.6. In the next release, the minimum will be 4.6.1.
- Development with Visual Studio 2012 or 2013. In the next release, the minimum will be Visual Studio 2015.
- Xamarin.Android development for API levels 15 (Android 4.0.3-4.0.4), 16 (Android 4.1-4.1.1), 17 (Android 4.2 - 4.2.2), and 18 (Android 4.3). In the future, the minimum will be API level 19 (Android 4.4).
- Xamarin.iOS development for iOS 9. In the next release, the minimum will be iOS 10.
- UWP and Xamarin.Forms UWP development for Windows 10 version 1607 (OS build 14393) and 1703 (OS build 15063). In the next release, the minimum will be version 1709 (OS build 16299), also known as the Fall Creators Update.

Copyright © 1995-2020 Esri. All rights reserved.

Issues resolved

- BUG-000092352: Japanese field names appear garbled when shapefiles with the Shift-JIS character code are loaded.
- BUG-000102308: `MobileMapPackage` doesn't honor complex symbology used in Utility Network. Requires mobile map package to be repackaged with ArcGIS Pro v2.1.
- BUG-000104759: The `RouteTask.CreateAsync` method fails to generate a route with the secured route service in UWP version 100.0. Note: Update Microsoft.NETCore.UniversalWindowsPlatform to v5.2.2.
- BUG-000105070: The `HitTest` method fails to return polyline graphics if the spatial reference of the map is a Geographic Coordinate System.
- BUG-000106470: `LayerCollection.Move()` method has no effect on the layer order.
- BUG-000107196: Formatted Label expression of Mobile Map Package isn't honored.
- BUG-000108002: Runtime does not use actual subtypes when populating `ArcGISFeatureLayerInfo.TypeIdFieldName` and `ArcGISFeatureLayerInfo.FeatureTypes` for a feature layer loaded from a mobile map package, unless the layer is symbolized on the subtype field. Requires ArcGIS Pro v2.1 (Q1 2018).
- BUG-000108493: Tif Rasters with a spatial reference of 102100 do not display in the correct location in a Runtime 100.1 application.
- BUG-000109265: Runtime SDK for .NET 100.1.0 guide states that we can use `ArcGISMap` class to create a map but this class is not there in the API reference.
- BUG-000109273: Authentication manager throws uncatchable null exception when user cancels authentication on Xamarin.Forms iOS.
- ENH-000100382: Provide a sample which demonstrates how to synchronize the viewpoints of the MapView and the SceneView.
- ENH-000103969: Functionality to use custom geotransformations missing in ArcGIS Runtime SDK for .Net 100.
- ENH-000107225: Support for Time on Layers and the MapView/SceneView.
- `DuplicateLabels.PreserveDuplicates` Not Respected in `GraphicsLayer`. See GeoNet discussion.
- Polygon graphic that crosses the dateline is clipped in dynamic mode . See GeoNet discussion.
- App freezes sometimes when zooming on S57 layer. See GeoNet discussion.
- `RouteTask.SolveRouteAsync` Error with curves and barriers. See GeoNet discussion.
- `System.ArgumentException` when using ShowCalloutForGeoElement. See GeoNet discussion.
- Polygons over poles not rendered correctly. See GeoNet discussion.
- `TextSymbol` being rendered incorrectly when passing across International Date Lines. See GeoNet discussion.
- `ServiceFeatureTable`: 'Invalid JSON' with Sqlite_error when adding a feature layer. See GeoNet discussion.
- App Crash While Adding Operational Layers. See GeoNet discussion.
- MMPK: Bad Symbol Rotation. See GeoNet discussion.
- `ShapefileTable` - How to project to a different spatial reference. See GeoNet discussion.
- MapView freezes permanently on iOS. See GeoNet discussion.

Copyright © 1995-2020 Esri. All rights reserved.

- App crashes frequently during zoom when many local feature layers are included in a map. See GeoNet discussion.
- MapView freezes permanently on iOS. See GeoNet discussion.
- ShapefileTable - How to project to a different spatial reference. See GeoNet discussion.
- TextSymbol HaloColor and HaloSize properties have no effect when used with graphics in dynamic mode in a SceneView.
- AGSMap's `initialViewpoint` does not work intermittently.
- Inconsistent drawing delay for some features from a service feature table when zooming in and out.
- Web map editing overrides are not applied to feature layers.
- Crash may occur on 5.1.1 Android Nexus 7 when zooming and panning with vector tiled layers.
- Reproject the area of interest to map's spatial reference when using `OfflineMapTask`.
- `TextSymbol` for polygon graphic spanning the dateline and rendered in static mode is displayed in the wrong location when zooming out.
- Z-aware feature layers do not render correctly in a `SceneView`.
- Feature layer loads but doesn't render when there is a "." in the field name.
- `Popup.GeoElement.Attributes` uses label/alias values instead of field names as keys for the attribute dictionary when used with ArcGIS map services.
- `RT_GeoView_getLayerViewState` fails for layer that has loaded but not rendered yet.
- Feature layers with indexes on GlobalID fields may cause an app to crash.
- Functions are not supported in where clauses for labeling.
- `ServiceFeatureTable` does not repopulate after calling `ClearCache`.
- Querying a feature table fails to call completion handler when querying a table that failed to load.
- Leaving location display running on a Windows or macOS desktop will crash after running for a short period of time.
- `AutoPanMode` and `InitialZoomScale` ignored if they are set before starting `LocationDisplay`.
- `GeoprocessingJob.ToJson` throws "invalid start of JSON" exception when it contains input parameter type of `GeoprocessingFeatures` with null features.
- `GeoprocessingJob.FromJson` does not populate `GeoprocessingFeatures.URL`. Cannot rely on deserialized `GeoprocessingJob` if it contains `GeoprocessingFeatures`.Subsequent serialization of job fail because it becomes an invalid JSON.
- Text in vector tiled layers may not display at certain scales.
- Models in a SceneView may render black on iPhone 5 devices.

Known issues

Layers

- Android: vector tile layers do not render on the Visual Studio emulator.
- BUG-000106101 Vector tiles using ArcGIS Runtime SDK for .NET fail to display some Chinese characters. Workaround: use a mobile map package (`.mmpk`) or a mobile basemap layer from a .geodatabase file.

Copyright © 1995-2020 Esri. All rights reserved.

- RasterLayer display issues if raster has non-square pixels (DTED, NITF).
- `RasterLayer` may only display when viewed at a large scale. Workaround: Use ArcGIS Desktop to build pyramids for the raster.
- `RasterLayer.Name` does not get populated with file name.
- `WMSSublayer.SubLayerContents` returns NULL instead of `ArrayObservable`.
- Vector tiled layers created from a URL do not have attribution.
- `ArcGISSceneLayer` does not support legend info.
- The following feature layer fields are not fully supported in ArcGIS Runtime: `subtypeField`, `defaultSubtypeCode`, and `subtypes` (these fields are new in ArcGIS Pro 2.1 and ArcGIS Enterprise 10.5).
- Loading feature layers with symbols that have an outline of less than 0 will return an exception.
- When adding rasters to a mosaic dataset that is already being used for a raster layer, the rasters will not display.

Maps (2D)

- Mobile map packages (.mmpk files) that are created in ArcGIS Pro 2.1 and that contain tile packages (.tpk files) or rasters are not supported.
- BUG-000103280 - While navigating a map on Android devices, basemap tiles do not load completely. Workaround: change the HTTP handler from `Default` to `Android`.
- BUG-000107699 - MapView.ShowCalloutAt doesn't place the callout at the defined MapPoint when running on Windows Phone. This bug only affects debug mode and is caused by an issue in the underlying JIT compiler for ARM.
- Identify operation on a WMS layer gives wrong results if the `MapView` is rotated.
- `MapView` `locationToScreen` returns incorrect coordinates in wrap around mode.
- `MapView` grid lines near the date line and poles may not display.
- Cloning an unloaded `Map` does not render in a `MapView`.
- Moving map view to a screen with a different DPI zooms the map in and out (5-10%).
- Map view doesn't zoom to geometry if the supplied geometry's envelope has zero height or width.
- Saving map with ForceSave false and feature collection layers by reference throws exception. Workaround: ForceSave the map and while loading the saved map set the visibility of FeatureLayer referenced by FeatureCollectionTable to true.

Scenes (3D)

- When setting a negative pitch value with the `rotateTo` function on `Camera`, roll is not changed.
- `Camera.rotateAround` does not apply the value for roll.
- For `Polygon` and `Polyline` graphic overlays, extrusion is not updated when value of attribute used in extrusion expression change.
- 3D `SimpleLineSymbol` arrow marker will display visible artifacts along the line when using static rendering.
- 3D `SimpleLineSymbol` arrow marker does not draw arrowheads when using dynamic rendering.

Copyright © 1995-2020 Esri. All rights reserved.

- Labeling features and graphics is not supported in a `SceneView`.
- Layer refresh interval is not supported on a `SceneView`.
- Graphics that are not visible can be identified in a `SceneView`.

Services

- When using `ExportVectorTilesJob` to download an item resource cache, job status will immediately change from `NotStarted` to `Succeeded`.
- Definition expressions using time fields on service feature tables will result in no features being displayed.
- Setting a color map raster function as rendering rule on an image service has no effect.
- Setting a rendering rule on `ImageServiceRaster` must occur prior to load.
- Identify does not return results when Geometry field is not exposed on service.
- Related tables: When an origin feature is deleted, the key field of the related feature is not set to null for hosted services that have a non-unique primary key.
- `OfflineMapSyncTask` doesn't always send updates to the service if it has layers with different origin spatial references.
- Loading a `ServiceFeatureTable` causes an invalid JSON error when the service contains a reserved field name, such as one that starts with `"gdb_"`.

Symbols, renderers, and graphics

- `PictureFillSymbol` ScaleX and ScaleY not supported in static rendering mode.
- Symbols in a layer do not render if the renderer has rotation expression based on a field that does not exist.
- Diagonal fill symbols render in opposite directions between JavaScript and Runtime.
- Symbol sizes for graphics in static mode differ in size between map and scene views.
- `TextSymbol` doesn't honor the screen alignment property for feature layers.
- Changing `TextSymbol` font weight or style is not supported on a `SceneView`.
- Font decoration, halo, and background properties are not supported on `TextSymbol` for graphics and features in dynamic rendering mode in a `SceneView`.
- Feature layers from hosted services on ArcGIS Enterprise 10.3 and 10.3.1 ArcGIS Servers cannot use advanced symbology.
- `LabelDefinition.ToJson` will return an empty array for unsupported text symbols.
- Saving a map with unloaded layers creates a portal item with no layers when displayed in ArcGIS Online.
- A Bing Maps layer created with a Portal needs to load the Portal first.
- New TypeKeywords and Tags objects are created when a `PortalItem` is loaded, rather than repopulating the existing objects.
- A service feature table will not render when unique value renderer contains classification values with commas.
- Well-formed polygons that intersect one of the globe's poles may not render correctly.
- Symbol rotation type (arithmetic/geographic) is ignored by graphics

Copyright © 1995-2020 Esri. All rights reserved.

- Creating a swatch from a scene symbol returns a null image.

Analysis

- Android: Line of Sight and Viewshed GeoAnalysis results do not render on the Visual Studio emulator.

Feature collections

- Cannot identify a `Point` feature with an ObjectId of 0 in a `FeatureCollection`.
- Z and M values in GeoPackage FeatureTables are not optional.
- `FeatureCollection.ToJSON` does not include popup info.
- When creating a feature collection from an array of graphics, the graphics' symbols are not honored. Instead, the default symbol is used.

Web maps

- Web maps with dynamic layers produced by ArcGIS Runtime do not draw correctly in ArcGIS Pro. Any overridden properties (renderer, labeling, opacity, sublayer visibility, and so on) will revert to service-specified defaults.
- Labeling behavior and display in a web map may not match between JavaScript and Runtime.

General

- WPF designer: Exception: Cannot create instance of "MapView" or Exception: Cannot create instance of "SceneView" error may be seen in the designer.
- Android: Application crash on Android 8 (Oreo) when using `Shared Runtime` and `Fast Deployment (debug only)`. Workaround: disable `Use Fast Deployment (debug only)`(and in some cases, you may also need to disable `Use Shared Runtime`) in `Project Properties > Android Options`. For more information, see https://bugzilla.xamarin.com/show_bug.cgi?id=60236.
- WPF: `Esri.ArcGISRuntime.WPF` cannot be used within an MFC-based project.
- Popup title does not use name when the title is defined with a coded value domain field.
- Field name in popup definition for Arcade expression does not include the title for the expression field.
- `Popup Symbol` returns null for features from a map service.
- When registering a mobile geodatabase with a new user (different from user who created the mobile geodatabase), local edits still apply the user who created the geodatabase when editor tracking is enabled.
- When using local street address locators, suggestion results without a house number cannot be used to retrieve results.
- Adding rasters to `MosaicDatasetRaster` with invalid parameters can fail without notifying the user.
- `FileNotFoundException` when opening incorrect file type for mobile map package (`*.mmpk`).

Related topics

Release notes
Release notes for 100.7.0

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.6.0
Release notes for 100.5.0
Release notes for 100.4.0
Release notes for 100.3.0
Release notes for 100.1.0
Release notes for 100.0.0

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.1.0

Version 100.1 of ArcGIS Runtime SDK for .NET, also known as Update 1, is the first update to version 100.0.

This topic describes what's new and changed in this release and provides a list of known issues. For the previous version's release notes, see Release notes for 100.0.

What's new

3D support on mobile devices commercially available

In the previous release, developers could deploy 3D apps to mobile phones and tablets only as BETA; 3D was commercially supported only on desktop platforms. Now at 100.1.0, 3D capabilities are fully supported on mobile phones and tablets.

Raster support on mobile devices commercially available

In the previous release, apps that used local raster data on mobile phones and tablets could only be deployed as BETA; Raster was commercially supported only on desktop platforms. At 100.1, local raster capabilities are fully supported on mobile phones and tablets.

Take a map offline

Web maps or specific areas of a web map from a portal can be downloaded and used in disconnected scenarios. The new `OfflineMapTask` provides a high-level API to download the map definition and its related data. When the map is taken offline, it is stored as a mobile map. If the map contains sync-enabled feature layers, those can be edited offline and the changes synced when network connectivity is later available. For details, see Offline.

Client-side labeling control

A new `LabelDefinition` class is introduced that defines how labeling is applied to feature data for a given scale range. You can specify the following:

- Which features are covered by the definition (by applying the label definition's where-clause)
- What text should be displayed, based on the individual feature's attributes and the label definition label-expression formula. Arcade expressions are supported in label definitions and visual variables but not in unique value renderers or class breaks renderers.
- How the text should appear, using the definition's text symbol
- Where the text should appear with respect to its feature, using the definition's label placement
- How to move or avoid other labels or features that would overlap each label

At this release, you can only construct a `LabelDefinition` instance from JSON but there is no explicit API for specifying labeling settings.

Support for additional layer types

This release adds support for the following data layer types.

- OGC WMTS
- Image Service
- Dynamic sublayers of an ArcGIS Map Service

Copyright © 1995-2020 Esri. All rights reserved.

- OpenStreetMap
- Bing

Extensible tile layers

This release provides APIs to display custom tiles. The `WebTiledLayer` is an extensible layer that requests map tiles from a web service based on a URI template. It provides a simple way to integrate non-ArcGIS Services as a layer in a map. Also, custom tiled layers can be created by sub-classing `ImageTiledLayer` and its subclass, `ServiceImageTiledLayer`. These two abstract classes expose protected abstract methods to either fetch tile images or return the URL of the tile images. By extending these two classes and implementing the abstract methods, you can create custom tiled layers and add them to a map.

Related tables

This release provides APIs for supporting related tables. With this, apps can handle workflows that involve reading, editing, and querying related data in both spatial and non-spatial tables. These related table workflows are supported in both online and offline maps. For details, see Relate features.

Display heat maps

You can now display heat maps from web maps.

Editing of public feature services available at the Lite license level

The ability to perform edits in a connected environment on a public feature services from ArcGIS Online or Enterprise at the Lite license level has been introduced this release. At version 100.0, all editing capabilities began at the Basic level. The editing of private feature services or editing in an offline environment remains at the Basic, Standard, and Advanced licensing levels.

Service area and closest facility analysis

Closest facility and service area tasks are now available for performing more network analysis in your ArcGIS Runtime applications.

A service area is a region that encompasses all streets that can be accessed within a given distance or travel time from one or more facilities. For example, a three-minute, drive-time polygon around a grocery store can determine which residents are able to reach the store within three minutes and are thus more likely to shop there.

The closest facility task generates routes between facilities and incidents. Routes may include a route shape and driving directions. The task may generate routes from facilities to incidents or from incidents to facilities and can find a user specified number of routes, for example finding the three closest fire stations for an address.

Coordinate formatting

The `CoordinateFormatter` class can be used to convert between points and formatted coordinates notation strings such as decimal degrees; degrees, minutes, and seconds; U.S. National Grid (USNG); and Military Grid Reference System (MGRS).

Follow a graphic in 3D

Camera controllers allow you to change the default camera behavior in a scene view. Two "orbit" camera controllers were added to lock the focus of a camera to a fixed point target or a possibly moving graphic.

Copyright © 1995-2020 Esri. All rights reserved.

Support for StreetMap Premium packages

StreetMap Premium for ArcGIS Runtime is a new data product, licensed as an extension to ArcGIS Runtime, that provides enriched street data, which powers a high-quality cartographic map and high-quality search, geocoding, and route analysis. Streetmap Premium for ArcGIS Runtime maps are consistent across all regions of the world and can be taken offline for disconnected use; they can simultaneously fulfill the need for an address locator, street network dataset, and basemap in your app.

StreetMap Premium for ArcGIS Runtime delivers HERE data as a mobile map package (an `.mmpk` file) for your app to access locally. This MMPK format allows the data to be accessed offline and therefore doesn't consume data from your user's data plan. This is the same high-quality data used by ArcGIS Online services, including the World Geocoding Service, Routing Service, and Street Map Service.

For more information, see Add StreetMap Premium data.

GeoView attribution

`MapView` and `SceneView` have been enhanced to automatically display attribution for map and scene layers respectively. The attribution is dynamic and updates as layers are added/removed, toggled on/off, or change visibility as they go in and out of scale range.

Issues addressed in this release

The following issues reported with ArcGIS Runtime SDK for .NET version 100.0.0 have been addressed in this release.

- Mobile Map Packages: cannot zoom or pan to the full extent of the data.
  - https://geonet.esri.com/thread/186550-rendering-mobile-map-packages
- Duplicate label behavior is not honoring the preserve duplicates setting.
  - https://geonet.esri.com/message/517477
- Marker symbols on lines created in ArcMap do not display correctly.
- Xamarin.Android: 'texture2D' type is for Vulkan api only.
  - https://geonet.esri.com/message/683700-net-runtime-sdk-100-errors-in-android-71
- `LocalServer` not returning all results when using `LIKE` and Wildcard operators.
  - https://geonet.esri.com/thread/188185-local-map-service-wildcard-query-not-working-in-quartz
- How to check DirectX Feature Level?
  - https://geonet.esri.com/thread/187265-how-to-check-directx-feature-level
- ClickOnce Deployments cannot locate native ArcGIS Runtime components.
  - https://geonet.esri.com/thread/186915
  - https://geonet.esri.com/thread/187613
- `GeodatabaseFeatureTable.AddAttachment` fails when unknown field is encountered.
  - https://geonet.esri.com/thread/189684
- DX11 Crash when rendering polygon on low-end hardware.
  - https://geonet.esri.com/message/661920
- `RuntimeDictionary.TryGetValue` returns always false.
  - https://geonet.esri.com/message/663767

Copyright © 1995-2020 Esri. All rights reserved.

- Calling `GraphicsOverlay.Graphics.Add(graphic)` raises exception: "Cannot call this method in this context: Graphic already has an assigned id."
  - https://geonet.esri.com/message/650536
- Crash on virtual machines when no hardware GPU support is present.
  - https://geonet.esri.com/thread/187265
- Failure to return results when performing an identify operation on `GraphicsOverlay`.
  - https://geonet.esri.com/thread/189597
- Correctly report Local Server license error when starting with **Lite** or **Basic** license.
  - https://geonet.esri.com/message/668835
- Graphics do not honor outline/border and selection when switching `RenderingMode`.
  - https://geonet.esri.com/message/664489
- Graphics do not honor outline/border and selection when switching basemaps with different SRs.
  - https://geonet.esri.com/message/664489
- Android: `SketchEditor` is `null` if `MapView` is in a predefined .axml layout file.
  - https://geonet.esri.com/thread/193203
- Android: frequent `SetViewpointAsync` calls causes an exception.
  - https://geonet.esri.com/thread/189153
- `RefreshObjectId` does not update feature attributes.
  - https://geonet.esri.com/thread/190364
- `PortalQueryParameters.CanSearchPublic` does not change a portal search.
  - https://geonet.esri.com/thread/192063
- **ENH-000085084**: Request to update local gdb with user's edits when `GeodatabaseSyncTask.SyncGeodatabaseAsync` is called.
- **ENH-000088684**: Move graphic anchor point (Rotate point) so it does not overlap with graphic vertex in .NET SDK.
- **BUG-000093112**: `GeodatabaseFeatureTable.QueryAsync` returns a bad allocation error when querying densely populated datasets.
- **BUG-000102890**: Undoing a previous edit that was made by draging a vertex using the SketchEditor class in the ArcGIS DotNet Runtime 100.0 SDK causes the vertex to disappear from the screen.
- **BUG-000093171**: PictureMarkerSymbol XOffset and YOffset do not work properly for the renderer of ArcGISDynamicMapServiceLayer in ArcGIS Runtime.
- **BUG-000103952**: Re-entering a Xamarin iOS application causes the map to (occasionally) render differently, stretching it either further down, or to the left on the screen.
- **BUG-000102348**: An unhandled exception is thrown when updating location in a map while the app is in background mode.
  - Also reported via https://geonet.esri.com/message/655273
- **BUG-000103092**: Rapid zoom/pan of a ArcGISTiledLayer crashes the application.
- **BUG-000082986**: `TextSymbol` doesn't honor `BackgroundColor` when `GraphicsLayer` `RenderingMode` is `Static`.
- **BUG-000103058**: Layers removed from an ArcGIS Online web map are still returned as OperationalLayers when loaded in ArcGIS Runtime Map.

Copyright © 1995-2020 Esri. All rights reserved.

- **BUG-000093873**: When multiple map views are used, the ArcGIS Runtime SDK for .NET app crashes in a local machine and freezes in the Remote Desktop Connection.
- **BUG-000102661**: Custom tiled layers cannot be added to a `SceneView` in ArcGIS Runtime SDK for .Net 100.0 applications.
- **BUG-000091175**: `LocalServer` returns `Error Code 500: Failed to create service` when loading a map package that uses third party fonts.

  Note:
  If you will run local services based on map documents that use third-party fonts, you must make sure that those third-party fonts are installed on each client machine on which your application will be run.

- **NIM097192**: Local GP Services do not handle Japanese characters in task parameter names.

Known limitations

The following list describes known limitations or issues with the 100.1.0 release.

Maps, layers, and general

- iOS: The minimum required version of **Xamarin.iOS** is **10.6.0.10**
- Projects created from the v100.0 ArcGIS Runtime Application (WPF) template require manual removal of the NuGet package error condition when upgrading to v100.1. Until the error is removed, the project will not build and reports `NuGet package Esri.ArcGISRuntime.WPF.100.0.0 is missing`. Follow these steps to remove the error.
  - In the Visual Studio Solution Explorer right click on the project.
  - Choose **Unload Project** from the content menu.
  - In Solution Explorer right click the unloaded project (listed as `unavailable`).
  - Choose **Edit .csproj / .vbproj** from the context menu to open the project file as XML.
  - Find and delete the following line (towards the bottom of the project file): `<Error Condition="!Exists('packages\Esri.ArcGISRuntime.WPF.100.0.0\build\net452\Esri.ArcGISRuntime.WPF.targets') And !Exists('..\packages\Esri.ArcGISRuntime.WPF.100.0.0\build\net452\Esri.ArcGISRuntime.WPF.targets')" Text="$([System.String]::Format('$(ErrorText)', '..\packages\Esri.ArcGISRuntime.WPF.100.0.0\build\net452\Esri.ArcGISRuntime.WPF.targets'))" />`
  - Close and save changes to the project file.
  - In the Solution Explorer right click the unloaded project.
  - Choose **Reload Project**.
- `Map.SaveAsAsync` and `FeatureCollection.SaveAsAsync` now take a `RuntimeImage` argument for the thumbnail rather than `System.Uri`. The overloads that take `System.Uri` are deprecated at this release. The easiest way to get a `RuntimeImage` for use as a thumbnail is to call `MapView.ExportImageAsync`.
- `MapView.ExportMapImage` doesn't work when executed in a Virtualized environment.
- `Layer.LoadAsync` intermittently fails to return to caller for some layer types when instantiated from `PortalItem`.

Copyright © 1995-2020 Esri. All rights reserved.

- **Workaround**: Instantiate the layer via URI (rather than a `PortalItem` instance), which is accessible on the `PortalItem`. The example below instantiates a `FeatureLayer` using this workflow.

```
var serviceItemUrl = "http://www.arcgis.com/home/
item.html?id=99fd67933e754a1181cc755146be21ca";
var serviceItem = await PortalItem.CreateAsync(new Uri(serviceItemUrl));

// Use the item's ServiceUrl instead of the item instance
var layer = new FeatureLayer(new Uri($"{serviceItem.ServiceUrl}/0"));

await layer.LoadAsync();
```

- When connecting to PKI-secured ArcGIS Portal for Enterprise, `ArcGISPortal.Uri` cannot be used directly to create a `CertificateCredential`. Runtime will continuously request authentication unless `/sharing` or `/sharing/rest` is included with the `CertificateCredential.ServiceUri`.
  - **Workaround**: The `ServiceUri` provided to `CertificateCredential` must be an absolute URI that includes `/sharing` or `/sharing/rest`.

```
// Define a credential request
CredentialRequestInfo cri = new CredentialRequestInfo
{
    // Define the service URI (include /sharing or /sharing/rest)
    ServiceUri = new Uri(@"https://myportal.com/gis/sharing/rest"),

    AuthenticationType = AuthenticationType.Token,
    GenerateTokenOptions = new GenerateTokenOptions
    {
        TokenAuthenticationType = TokenAuthenticationType.OAuthImplicit
    }
};
```

- When iterating a `FeatureQueryResult` (using `foreach`, e.g.), asynchronous calls throw an exception.
  - **Workaround**: iterate results using `FeatureQueryResult.ToList()`, `FeatureQueryResult.ToArray()`, or `FeatureQueryResult.ElementAt(index)`.
- A call to `Feature.LoadAsync` when looping through `RelatedFeatureQueryResult` causes an extra iteration.
  - **Workaround**: iterate results using `RelatedFeatureQueryResult.ToList()`, `RelatedFeatureQueryResult.ToArray()`, or `RelatedFeatureQueryResult.ElementAt(index)`.
- Android: Vector tile layers do not render on the Visual Studio emulator.
- Xamarin Forms: `MapView` or `SceneView` will not display if placed in a `Grid.Row` that has its `Height` defined as `Auto`.
- When using `LocalGeoprocessingService` in a Local Server with `GeoprocessingServiceType.AsynchronousSubmitWithMapServiceResult` mode, `GeoprocessingResult.MapImageLayer` isn't populated correctly.
  - **Workaround**: build a URL that references the map image result and use it to create a new `ArcGISMapImageLayer`.

```
// ** Workaround for local server when using result map server to render results as an image
// Note: the 'job' variable represents a GeoprocessingJob that has completed

// Get URL of the local geoprocessing service from LocalGeoprocessingService
var serviceUrl = localGeoprocessingService.Url.ToString();
```

Copyright © 1995-2020 Esri. All rights reserved.

```
// Construct URL to the map server result
var mapServerUrl = serviceUrl.Replace("GPServer", $"MapServer/jobs/{job.ServerJobId}");

// Create new ArcGISMapImageLayer from the constructed URL
var mapImageLayer = new ArcGISMapImageLayer(new Uri(mapServerUrl));

// ** Workaround end
```

- `MapViewLocationToScreen` gives incorrect coordinates in wrap around mode.
- Map view grid lines that lie close to the 180th meridian or poles may not always display correctly.
- AutoPanMode and InitialZoomScale are ignored if set before starting location display.
- Querying `FeatureTable` does not complete if the table failed to load.
- Cloning an unloaded map prevents it from rendering in a map view. Load the map before cloning it.
- Text in vector tiled layers may not display at certain scales.
- Vector tiled layers created from a URL don't have attribution.
- Once a vector tile layer is cloned, there is no way to handle requests for it.
- Related tables: When an origin feature is deleted, the key field of the related feature is not set to null for hosted services that have a non-unique primary key.
- Web maps saved from an ArcGIS Runtime app containing Bing Maps layers may not be compatible with older versions of the Web Viewer.
- ArcGIS Online currently does not support requesting features from feature services in a different spatial reference using the latest WKID value stored in a map.
- SpatialReference.equals() may return false when comparing Web Mercator spatial references.
- When exporting a tile package from a service that is hosted in ArcGIS Online (tiles.arcgis.com) with a geometry that is not in the same spatial reference as the service, the tile package will be corrupted and will not display correctly.
- `ArcGISSceneLayer` does not support legends.
- `GeoView.GetLayerViewState` fails for a layer that has not finished rendering.
- Using `SketchEditor`, you cannot digitize a polyline across the dateline.
- Memory is not freed when graphics symbolized with picture marker symbols are cleared from a graphics overlay.
- A Bing Maps layer created with a Portal needs to load the Portal first.
- Definition expressions using time fields on spatial service feature tables will result in no features being displayed.
- Cannot change definition expression after setting invalid definition expression on feature layer in a SceneView.
- Feature layers with indexes on GlobalID fields may cause an app to crash.
- Loading feature layers that have fill symbols with an outline with of less than 0 will return an exception.
- Diagonal fill symbols render in opposite directions between JavaScript and Runtime.
- Moving map view to a screen with a different DPI zooms the map in and out (5-10%).
- Moving map view to a screen with a different DPI changes the size of the map content.

Copyright © 1995-2020 Esri. All rights reserved.

- Symbol sizes for graphics in static mode differ in size between 2D and 3D views.
- Vector tile symbols don't always scale consistently with DPI.
- Map view doesn't zoom to geometry if the supplied geometry's envelope has zero height or width.
- Setting minimum scale on a graphics overlay in a scene view is not accurate.
- Calling the clone method on a loaded feature layer throws `bad_weak_ptr` exception.
- Gradient fills used in mobile map package are not displayed correctly in apps.
- Invalid base layer, reference layer, or operational layer will not raise `LayerViewStateChangedEvent` under certain scenarios.
- In some cases, a feature layer can have a load status of `Loaded` but `Renderer` still returns null.
- Service feature table contents do not re-populate after clearing cache.
- Service feature table can fail to load from a service with "Invalid JSON" error when the real problem is the JSON from the service contains an unsupported image format.
- There's an inconsistency in handling null values in JsonSerializable UnknownJson.
- Popup title does not use name when the title is defined with a coded value domain field.
- Popup.GeoElement.Attributes uses label/alias values instead of field names as keys for the attribute dictionary when used with ArcGIS map services.
- Field name in popup definition for Arcade expression does not include the title for the expression field.
- `Popup Symbol` returns null for features from a map service.
- A geodatabase is taken out of scope even if its feature tables are in scope. This is because the geodatabase holds on to strong references to its tables, but the tables do not have a strong reference back to the geodatabase (as this would lead to a circular reference). This is by design. It's recommended to keep the geodatabase as a global variable.
- Editing overrides for a feature layer in a web map are not honored.
- Synchronizing large geodatabases with many features and no local edits may take longer than expected.
- Offline features disappear intermittently during map interaction when syncing large geodatabase.
- When registering a mobile geodatabase with a new user (different from user who created the mobile geodatabase), local edits still apply the user who created the geodatabase when editor tracking is enabled.
- When using local street address locators, suggestion results without a house number can't be used to retrieve results.
- Locator performance may be poor for larger local locator datasets.
- Saving map with ForceSave false and feature collection layers by reference throws exception. Workaround: ForceSave the map and while loading the saved map set the visibility of FeatureLayer referenced by FeatureCollectionTable to true.
- Functions are not supported in where clauses for labeling.
- GeoprocessingJob.ToJson throws exception when it contains input parameter type of GeoprocessingFeatures with null features.
- GeoprocessingJob.FromJson does not populate GeoprocessingFeatures.URL.

Raster

- Adding rasters to `MosaicDatasetRaster` with some invalid parameters can fail without notifying the user.

Copyright © 1995-2020 Esri. All rights reserved.

- Some 3D renderers on some platforms may not work correctly on `MosaicRasterDataset`.
- In some cases, the default rendering applied to raster data may appear different from how it appears in ArcGIS Online.
- The `LoadStatusChanged` event doesn't fire when using a `MosaicDatasetRaster` instance to create a mosaic dataset and binding it to a `RasterLayer` for rendering.
- When adding rasters to a mosaic dataset that is already being used for a raster layer, the new content will not show up in that existing layer.
- Raster layers report `LayerViewState.Error` ("Layer does not have SpatialReference") while loading. The layer will then transition to `LayerViewState.Active` after self-correcting and loading the layer.
- `RasterLayer` doesn't free a used raster before the application is shut down.
- Xamarin Android: Some raster formats crash the app when deployed on VisualStudio Emulator.

Image services

- When an `ImageServiceRaster` is created with a URL, `Path` should return null but instead it returns the URL.
- Setting a color map raster function as rendering rule on an image service has no effect.
- Visual artifacts can appear on a map when the default rendering rule of an image service has dynamic range adjustment (DRA) enabled.
- Setting a rendering rule on `ImageServiceRaster` after loading has no effect.
- App may crash when exporting tile cache from image service that does not support it.

Symbols, renderers and graphics

- `TextSymbol` doesn't honor the screen alignment property for feature layers.
- Kerning does not work with simple Latin scripts, causing a performance problem with military symbol text.
- `PictureFillSymbol` width, height, and image properties are not working correctly on scene view.
- When kerning is enabled on a statically-rendered `TextSymbol`, it will appear misaligned and may render beyond the extent of its background.
- In a scene view, changing `TextSymbol` font weight or style does not affect the symbol as expected. The glyph spacing changes, but the glyphs themselves remain the same.
- In a scene view, changing `TextSymbol` halo color or size has no effect when symbolizing elements displayed in dynamic mode.
- In a graphics overlay displayed using dynamic mode in a scene view, applying a font decoration to `TextSymbol` has no effect.
- Crash when `DrawingInfo` uses field alias name instead of field name when using advanced symbology.
- Feature layers from federated 10.3 and 10.3.1 ArcGIS Servers cannot use the default advanced symbology setting.
- When a specific offset value is applied to both dynamic and static picture marker symbols, the displacement is different on both the graphics.

Copyright © 1995-2020 Esri. All rights reserved.

- In static rendering mode, setting Y offset on `TextSymbol` causes the symbol to render twice, once at the old and once at the new location.
- When invoking `ToJson` on `LabelDefinition`, certain text symbols may get serialized as "{}", losing their properties.
- Symbols should be loaded before being used in a renderer sent to a service.
- Random tile flickering may occur when adding polygons to a graphic overlay with horizontal fill symbol and transparent color, in static rendering mode.
- `TextSymbol` for a polygon graphic spanning the dateline and rendered in `Static` mode is displayed in the wrong location when zooming out.
- TextSymbol HaloColor and HaloSize properties have no effect when used with graphics in dynamic mode in a SceneView.
- Military polyline and polygon symbols that render correctly in 2D do not render correctly in a graphics overlay in dynamic mode in 3D.
- Military symbols requiring geometry conversions where the data is not multipoint are drawn incorrectly when zooming out.
- Some military line symbols do not display in the correct location.
- Symbol rotation type (arithmetic/geographic) is ignored by graphics.
- Creating a swatch returns a null image for scene symbols.
- Extruded graphics in a scene view may disappear while navigating the map.
- Selection halo thickness for various types of graphics are not consistently sized.
- Selection color does not display for a selected polygon with a transparent outline in a dynamic graphics overlay in a SceneView.
- Graphics overlays drawn in static mode do not render correctly. Use graphics overlays in dynamic mode.
- A service feature table will not render when unique value renderer contains classification values with commas.

Portal

- New TypeKeywords and Tags objects are created when a `PortalItem` is loaded, rather than repopulating the existing objects.

Feature collections

- A feature collection in a web map that doesn't have a spatial reference for its extent will not draw.
- `FeatureCollection.ToJSON` loses popup info properties.
- Creating `FeatureCollectionTable` from `FeatureQueryResult` can throw an exception.
- When creating a feature collection from an array of graphics, the graphics' symbols are not honored. Instead, the default symbol is used.
- Very large feature collections may cause an app to run out of memory and crash.

Web maps

- Web maps with dynamic layers produced by ArcGIS Runtime do not draw correctly in ArcGIS Pro. Any overridden properties (renderer, labeling, opacity, sublayer visibility, and so on) will revert to service-specified defaults.

Copyright © 1995-2020 Esri. All rights reserved.

- Labeling behavior and display in a web map may not match between JavaScript and Runtime.

3D

- Scenes do not support vector tiled layers or image services.
- Files added to an existing raster elevation source at runtime are not applied to the scene view.
- Elevation of vertices of a 3D polyline drawn high above the surface, viewed in relative mode, can dip when the tile beneath it is out of view and therefore unloaded.
- `SceneView.ScreenToLocationAsync` can return inaccurate Z values.
- The rotation property of a viewpoint is not applied when it is used to set a viewpoint for another view.
- The color of a `ModelMarkerSymbol` can't be changed.
- Models in a SceneView may render black on Android simulators.
- Z-aware feature layers do not render correctly in scene view.
- Xamarin Forms: Applying a `BlendRenderer` to a `Raster` displayed in a `Scene` crashes when running on iOS or Android.
- When using a scene layer package with a terrain, it can be seen through terrain.
- Scene layer may not display due to limited device memory.
- Alpha property incorrectly applied during image export of a SceneView.
- Labels for feature layers do not display on scene view.
- An application might fail when using military symbology layers in a graphics overlay with a scene. This does not happen with military feature layers.
- Setting minimum scale on a graphics overlay in a scene view may not always be accurate.
- Set viewpoint with NAN scale value causes globe in SceneView to render black.
- Edges of polyline may appear behind globe in SceneView.
- A selection highlight doesn't work when two graphics overlap.
- `ScreenToLocationAsync` sometimes gives unexpected elevations when zoomed out.
- In a scene view, changing the identify tolerance for graphics overlay behaves as though a zero tolerance is set.
- Graphics overlay selection halo does not appear around selected symbols that are fully transparent.
- The thickness of the selection halo varies between different symbol types.
- For both compact and exploded caches in Web Mercator, an error occurs reading the cache configuration.
- Rendering graphics in static mode near the poles or at varying distances from a camera may display unexpected results.
- Camera from CurrentViewpoint on SceneView always returns null.
- Multiple touch points for scene interactions, except for zoom, have not been implemented yet.
- Layer refresh interval does not work in 3D scenes.
- When rotating down on a scene view, the rotation stops when the clicked point is at 0,0.
- For a `Scene` (3D), WGS 1984 is the only supported spatial reference.

Copyright © 1995-2020 Esri. All rights reserved.

- App can crash when no symbol is set on a extrusion renderer.
- Portions of building models may become transparent when panning the 3D scene.
- Models in a SceneView may render black on iPhone 5 devices.
- Identify operation can return empty results for `ArcGISSceneLayer`.
- Graphics that not visible can be identified in a SceneView.
- Unable to identify a partially-transparent symbol for a graphic in a dynamic graphics overlay in a SceneView.
- Visibility of a graphic in a SceneView is determined by geometry, not display location.
- Transparent simple marker graphics on a 3D scene view do not correctly display all symbols that are directly behind them.
- A 3D scene may exhibit significant flickering while adding a new feature.
- Selected features may flicker in a scene view.
- Scene layer packages are sometimes visible through the terrain.
- 3D vertical extrusion is applied to a graphic's symbol when it should only be applied to the renderer's symbol.
- Rendering of dynamic graphics in a SceneView is fragmented in iOS simulator.

Related topics

Release notes
Release notes for 100.7.0
Release notes for 100.6.0
Release notes for 100.5.0
Release notes for 100.4.0
Release notes for 100.3.0
Release notes for 100.2.0
Release notes for 100.0.0

Copyright © 1995-2020 Esri. All rights reserved.

Release notes for 100.0.0

This major release of ArcGIS Runtime SDK for .NET introduces new and changed APIs from previous releases. The SDK provides an enhanced Windows Presentation Foundation (WPF) API for building desktop applications for Windows and a new Universal Windows Platform (UWP) API for building apps for Windows 10 desktops and devices. It also includes APIs for developing native iOS and Android apps as well as cross-platform Xamarin Forms apps using Visual Studio and C#.

Note:
In the previous (beta) release, the Xamarin APIs were in a separate SDK called **ArcGIS Runtime SDK for Xamarin**. This SDK no longer exists as a separate product, but all of its functionality has been incorporated into **ArcGIS Runtime SDK for .NET**.

The 100.0 release is the first release in the 100.0.0 family. The 100.0.0 code name reflects the next generation of ArcGIS Runtime, built from the ground up using a new architecture. This has enabled new capabilities and performance optimizations and provides API and functional consistency across all of the Runtime SDKs. The entire API was re-imagined to incorporate these new capabilities and best-of-breed ideas from all of the ArcGIS Runtime SDKs.

This topic describes what's new, what's changed, and known limitations in this release.

What's new

This release of the ArcGIS Runtime puts the map and the scene at the heart of the API to allow developers to quickly leverage the ArcGIS platforms Web GIS capabilities. The following additions highlight the benefits of the new ArcGIS Runtime release.

Maps and scenes

The Map and the Scene objects have been split out from the views that display them, to represent model components from a Model-View-Controller architecture. These important objects at the heart of the ArcGIS Runtime have an API that follows the ArcGIS Web GIS information model. They contain operational layers, basemaps, bookmarks, popups and other ArcGIS specific data to enable these to be leveraged in your apps. They also include convenient APIs to instantiate them from URLs, portal items or default basemaps so you can get great maps working quickly.

Note:
Version 2.0 of web maps and higher are supported (web maps from Portal for ArcGIS 10.3 and above); pre-2.0 versions are no longer supported.

Views

GeoViews (MapView and SceneView) are solely responsible for display and interaction, separating concerns from the model objects and allowing the APIs to be simplified and harmonized between the 2D and 3D worlds. The Views contain Graphic Overlays, as well as operations to easily identify features and graphics without having to write any layer type specific code.

Go offline

To make things easy for you, the APIs for common operations such as editing, searching, geocoding or routing are the same whether you are online or offline.

- **Mobile map packages —** Offline maps with great symbology, place search and directions can be packaged with ArcGIS Pro, side loaded onto your device and used in the Runtime through Mobile Map Packages. A mobile map package is a set of

Copyright © 1995-2020 Esri. All rights reserved.

items bundled together for transport, often for use in offline workflows. The items are one or more maps, their associated layers and data, and optionally networks and locators. A mobile map package also includes metadata about the package.

- **Offline services —** You can also take your connected ArcGIS based feature and tiled layers offline line on demand with dedicated Tasks and associated Jobs. The GeodatabaseSyncTask works with ArcGIS feature services to take features offline in a mobile geodatabase (.geodatabase file) and allow them to be edited and synced. The ExportTileCacheTask extracts tiles from a tiled ArcGIS map service as a tile package (.tpk file) and allows them to be viewed offline.

For more information, see the "Create an offline map" topic in this guide.

3D

On desktop platforms, 3D has been brought to the ArcGIS Runtime for native app development. Build 3D scenes with raster, tiled and vector data sets including 3D specific symbology for great "better than flat" visualizations. This capability is Beta on mobile platforms.

For more information, see the "Display a scene" topic in this guide.

Vector tiled layers

You can use vector tiled layers in your ArcGIS Runtime app. These layers are similar to raster-based tiled layers but they are faster, smaller, and look better on high resolution devices. They also differ in the way cartography is delivered. Instead of raster pixels, vector tiled layers deliver cartography using vectors so cartography is rendered at runtime leveraging all the available pixels on the device. The file format is binary and conforms to the Mapbox vector tile specification. Vector tiled layers don't contain any feature data and they don't support identify or search operations. For details, see the API reference for `ArcGISVectorTiledLayer` or see the "Tiled layers" section in the "Layers" topic in this guide.

Graphics as overlays

Graphics have always been used to display temporary application data on top of a map. To formalize this usage, graphics are now added to GeoViews (MapView or SceneView) as overlays. This ensures that graphics are always displayed on top, even when map layers are reordered. This also makes it convenient to switch out maps while still keeping the graphics in place. For more information, see the "Add graphics overlays to your app" topic in this guide. It is now also possible to identify graphics overlays in a map view via a single convenient method.

Raster

This capability is Beta for deploying on mobile platforms.

You can now add raster data on desktop devices from many popular raster file formats such as DTED (Digital Terrain Elevation Data), GeoTIFF, RPF (Raster Product Format), NITF (National Imagery Transmission Format), HRE (High Resolution Elevation), Erdas Imagine, etc.

World files are no longer required to display GeoTiff and NITF formats.

Change the visualization of raster data by computing hillshades, applying colormaps, or stretching the statistical distribution of pixel values.

New layer names

Other changes have been made to layer names: Use ArcGIS map image layers for adding ArcGIS Server map services. Use ArcGIS tiled layers for adding ArcGIS Server tiled map services and tile packages. Use a Feature layer with associated Feature

Copyright © 1995-2020 Esri. All rights reserved.

Tables for adding layers or tables from ArcGIS feature services and geodatabases (including non-spatial tables). See the "Layers" topic in this guide for information on the supported layer types.

Web map pop-ups

Web map pop-ups allow you to customize how the data in your layers are presented to users. A new full popup API has been added to allow you to tap into the their configuration and build great user experiences for your users which include feature editing. Those layers which allow popup definitions implement the new PopupSource interface. The map view also provides coarse-grained methods to identify pop-ups taking into consideration different types of layers, their visibility, scale range, and pop-up configuration, and so on, and returns a list of pop-ups for features at a given location.

Note:
Version 2.0 of web maps and higher are supported (web maps from Portal for ArcGIS 10.3 and above); pre-2.0 versions are no longer supported.

Portal integration

The mapping API seamlessly integrates with the portal API, allowing you to access portal content and use them as maps. You can also edit existing maps or author completely new maps and save them back to the portal, which can then be used elsewhere in the ArcGIS system. For details see the "Access portal content" topic in this guide.

Military symbology

On desktop platforms, the API and workflow for using Military symbology has been greatly simplified in this release. You can now use military dictionary style files as the basis for a Dictionary Renderer which can be applied to graphics overlays OR feature layers. This capability is Beta on mobile platforms.

Loadable pattern

Resources, such as maps, layers, tasks etc, that load metadata asynchronously to initialize their state adopt the loadable pattern. This pattern makes the behavior of loading state more consistent and uniform, and also makes its asynchronous nature more explicit. Loadable resources do not automatically load their state. They load lazily, when asked either by the developer, or by other objects that depend on it. The status of a loadable resource can be easily monitored to determine if it is loading, loaded successfully, or failed to load, and you can retry loading it if it failed to load. For more information, refer to the loadable pattern topic in this guide.

Loadable features

Features retrieved from `ArcGISFeatureTable` have been optimized to contain only a minimum set of attributes which are required to render them on a map. To get the full list of attributes, the feature has to be explicitly loaded. This optimization reduces latency and bandwidth consumption while displaying features, but still allows you get the feature's complete data if you want to edit or display feature details. When editing features, they must be loaded first.

Centralized handling of authentication

All security and authentication related aspects are managed by a new`AuthenticationManager` class which helps to unify and centralize how authentication is performed regardless of the security mechanism being used or the operation being performed. The authentication manager issues an authentication challenge whenever security related events are encountered. Developers

Copyright © 1995-2020 Esri. All rights reserved.

can monitor these challenges and respond with credentials to get access to secured resources, or allow the authentication manager to prompt the end user for credentials. For more information, refer to the API reference documentation for `AuthenticationManager`

Geometries and geometry builders

The fundamental geometry objects remain the same - point, multipoint, polyline, and polygon but they are immutable and cannot be modified once they are created. The constructors on geometry objects allow them to be created easily in one shot if their coordinates are known, but you can also use geometry builders to progressively construct new geometries, either from scratch or by using an existing geometry as the starting point and then manipulating its coordinates. For more information see the "Edit geometries" topic in this guide.

Error handling

It's now easier to determine where errors occur in the stack so you can provide better error messages to your users. A new standardized error domain property is available, which indicates if the error occurred client-side within the ArcGIS Runtime or server-side from an ArcGIS Server service or web service. A consistent error code property can be used to further diagnose what the error was and what error message should be displayed to the user. A listing of error code descriptions is provided in the "Platform error codes" topic in this guide.

Changes since the .NET and Xamarin betas

If you participated in beta testing for ArcGIS Runtime SDK for .NET or ArcGIS Runtime SDK for Xamarin, you will notice the following changes.

- ArcGIS Runtime SDK for .NET includes APIs for WPF and UWP plus the APIs previously released as ArcGIS Runtime SDK for Xamarin.
- The API reference for Xamarin.Forms is no longer a separate reference. In this release, it is included with Xamarin.Android, Xamairn.iOS, and UWP API Reference.
- KML Layer has been removed and will be reintroduced in a future update.
- ArcGIS Runtime SDK for .NET is now installed using Visual Studio Installer (.vsix) technology (as opposed to the previous Windows Installer (.msi) setup).
- APIs for WPF and UWP are now delivered as NuGet pakages instead of assembly and extension SDK references.
- Some system requirements have been updated. See the System requirements topic for details.
- Deployment and licensing support has been added.

Known limitations

The following list describes known limitations or issues with the 100.0 release. Where one is available, a workaround is described.

- `MapView.ExportMapImage` doesn't work when executed in a Virtualized environment.
- When iterating a `FeatureQueryResult` (using `foreach`, e.g.), asynchronous calls throw an exception.
  - **Workaround**: iterate results using `FeatureQueryResult.ToList()`, `FeatureQueryResult.ToArray()`, or `FeatureQueryResult.ElementAt(index)`.

Copyright © 1995-2020 Esri. All rights reserved.

- In a **Universal Windows Platform (UWP)** app, properties that require a URI (Layer.Source, e.g.) cannot be set in XAML if running on `Windows 10 Build 10240`.
  - **Workaround**: Upgrade to `Windows 10 Version 1511 Build 10586` or later.
- In a **Universal Windows Platform (UWP)** project using **NetCore 5.2.2** an exception is raised ("No parameterless constructor defined for this object") when calling `ArcGISPortal.CreateAsync`.
  - **Workaround**: Use the previous release of the `NetCore` serialization packages by adding the appropriate dependencies in the **project.json** file. See more details in this GitHub issue.
- When using `LocalGeoprocessingService` in a Local Server with `GeoprocessingServiceType.AsynchronousSubmitWithMapServiceResult` mode, `GeoprocessingResult.MapImageLayer` isn't populated correctly.
  - **Workaround**: build a URL that references the map image result and use it to create a new `ArcGISMapImageLayer`.

```
// ** Workaround for local server when using result map server to render results as an image
// Note: the 'job' variable represents a GeoprocessingJob that has completed

// Get URL of the local geoprocessing service from LocalGeoprocessingService
var serviceUrl = localGeoprocessingService.Url.ToString();

// Construct URL to the map server result
var mapServerUrl = serviceUrl.Replace("GPServer", $"MapServer/jobs/{job.ServerJobId}");

// Create new ArcGISMapImageLayer from the constructed URL
var mapImageLayer = new ArcGISMapImageLayer(new Uri(mapServerUrl));

// ** Workaround end
```

- Navigation can appear slow when a debugger is attached and the map has the accelerated display mode enabled. The display can appear to "stutter" when zooming in, especially on tile layers.
  - **Workaround**: The issue is due to the performance impact of the debugger interacting with .Net exception handling. The exceptions are within the .Net framework and are caused by cancelling in-progress web requests for tiles that are not needed for display. It is therefore recommended to verify application performance in release mode (without a debugger attached).
- `SceneView.ElevationExaggeration` has no effect on Xamarin APIs.
- In an iOS app, touch events do not propagate to a child control added as a sub-view of a `MapView` control. This may happen without your knowledge when you drop a view on top of the map in the designer, in which case it gets added as a sub-view of the map view. There are a few ways to avoid or workaround the issue.
  - **Workaround**: Move a view that has already been added in this way up the visual hierarchy so that it's not a child of the map view.
  - **Workaround**: When dragging the view to the design surface, drop it somewhere that does not overlap the map view. This will avoid it being created as a child.
  - **Workaround**: Add the view programmatically.
- Types that derive from `Esri.ArcGISRuntime.RuntimeObservableCollection<T>` have the following limitations.
  - An instance of an object cannot be added to the same collection twice.
  - An instance of an object cannot be added to multiple collections.
  - Null object instances cannot be added to the collection.

Copyright © 1995-2020 Esri. All rights reserved.

  - Existing instances in the collection cannot be set to null.
- `Esri.ArcGISRuntime.Xamarin.Android` and `Esri.ArcGISRuntime.Xamarin.Forms` do not currently support **arm64-v8a** as a target CPU architecture for Android apps. Apps configured to support **armeabi-v7a** can run on 64-bit ARM devices. Support for `Xamarin.Android` on **arm64-v8a** is on the roadmap for a future release.

Maps

- Deadlocks may occur if a map is destructed while it has outgoing requests.
- In some cases, wrap-around does not stay enabled.
- Map zooms in and out (5-10%) when dragged between monitors with different DPI
- Some specialized layers are not supported in this release: KML, WMS, WMTS, OpenStreetMap, Bing, CSV by reference and GeoRSS.
- Maps do not open consistently at exactly the same initial extent.
- Map view grid lines close to the date line and poles may not display.
- MapView doesn't zoom to geometry if the supplied geometry's envelope has zero height or width.
- Cloning/copying an unloaded map does not render in a map view.
- Quickly opening a mobilemap and then destroying the mapview may cause a crash

Web maps

- A web map won't open when a feature layer's type ID is not an integer.
- Pop-up title does not use name when its defined with a coded value domain field.
- There's an inconsistency in handling null values in JsonSerializable UnknownJson.
- Map does not display any features for FeatureLayer with date-based definition expression using the timestamp keyword.

Layers

- Vector tiled layers (ArcGISVectorTiledLayer)
  - DrawStatus does not complete when zoomed out to lowest level of detail (LOD) and no labels are displayed
  - Vector tile symbols don't scale consistently with DPI
  - Vector tiled layers created from a URL don't have attribution
  - Crash may occur on Android when zooming and panning with vector tiled layers.
  - A vector tiled layer may crash if you interact with it before it has begun drawing.
- ArcGISTiledLayer fails to load with cached image service as portal item
- If the URL property under basemap doesn't exist in a layer type unsupported by ArcGIS Runtime, the layer will not get serialized.
- Overrides to definition expressions on sublayers in a map image layer from a portal item are not respected.

Symbols, renderers and graphics

- Points are not drawn on a graphics overlay if they're on the dateline.

Copyright © 1995-2020 Esri. All rights reserved.

- If min/max scale is changed on graphics overlay before the map is loaded, min/max scale on the map view is locked.
- The LocationDisplay symbol may sometimes behave strangely in navigation mode.
- ClassBreaksRenderer with one class break and no minimum value symbolizes features incorrectly
- `Popup.Symbol` returns null for features from a map service.
- For GraphicsOverlay, the selection doesn't show for fully transparent symbols.
- When creating a feature collection from an array of graphics, the graphics symbols are not honored.
- For graphics overlays in dynamic mode, a renderer's rotation is used even if the graphic's symbol is overridden.
- Symbol rotation type (arithmetic/geographic) is ignored by graphics.
- Extrusion is applied to symbol in graphics when it should only be applied on symbol in renderer
- Feature layers from federated 10.3 and 10.3.1 ArcGIS Servers cannot use the default advanced symbology setting.
- Features with uncompressed BMP symbols may not display.
- Static graphic overlays render fill symbols with an outline even if it is set to Null. Use a transparent outline instead.

Feature layers

- Large amounts of local feature data delays tiles
- When syncing a large geodatabase, offline features disappear intermittently during map interaction
- Synchronizing large geodatabases with many features and no local edits may take longer than expected.
- ArcGIS Online doesn't support requesting features from feature services in a different spatial reference using the latest WKID value stored in a map
- TextSymbol doesn't honor the screen alignment property.
- Creating a feature with an invalid field name throws an incorrect error (an ownership-based access error)
- Sometimes, when changing zoom levels, there's a drawing delay for symbols for features in a FeatureLayer.

3D

- Vector tile layers are not supported on 3D scene views. These are only for 2D viewing in this release.
- SceneView doesn't render on Galaxy s3
- Graphics overlays in scene views
  - Graphics overlays drawn in static mode do not render correctly. Use graphics overlays in dynamic mode.
  - 3D Symbols are not supported on Graphics or in GraphicsOverlay Renderers when the GraphicsOverlay.RenderingMode property is set to GraphicsRenderingMode.Static.
  - Setting min scale on a graphics overlay is not accurate
  - Selection halo thickness for graphicsare not consistently sized.
  - Extruded graphics in a scene view may disappear while navigating the map.
  - Military lines and polygons don't render correctly in dynamic mode in 3D.
  - Symbol sizes for graphics in static mode differ in size between 2d and 3d

Copyright © 1995-2020 Esri. All rights reserved.

- toJson doesn't work with scene symbols.
- 3D feature symbol size is not always consistent with 2D.
- Creating a swatch returns a null image for scene symbols.
- Elevation sources added to a scene at runtime are not applied to the scene view.
- Elevation sources from local raster files may crash on Android.
- Callouts and attribution are not visible in a scene view on macOS.
  SceneView.ScreenToLocation returns innacurate Z values
- The Identify operation:
  - On GraphicsOverlay does not return results for text symbols when using a 3D scene view.
  - On the GraphicsOverlay method ignores the search tolerance.
  - Returns empty results for ArcGISSceneLayer.
  - On macOS sometimes does not return results from map image layer sublayers
- The sun's position doesn't change with different values for the `sunDateTime` variable on a scene view.
- Scene layer transparent pixels are not rendering correctly on Android platforms.
- Transparent simple marker graphics on a 3D scene view do not correctly display all symbols that are directly behind them.
- SceneView screenToLocationAsync may provide inaccurate elevation results at small scales.
- Feature layers displayed on 3D scene views do not render in the ON_INTERACTION_CACHE mode. When using scene views, the feature layers should be used in MANUAL_CACHE mode.
- Updating graphics with ModelSceneSymbol and scene views in the map at the same time causes flickering.
- Converting screen coordinates to location asynchronously for 3D scene views can generate inaccurate results when the elevation is not fully loaded. This can occur when the camera is very far from the surface.
- Changing the opacity of a graphics overlay on a 3D scene view does not automatically take effect. It only updates following a pan or zoom interaction.
- SimpleMarkerSceneSymbol, DistanceCompositeSymbol, and ModelMarkerSymbol each returns an empty string when you try to get its JSON representation.
- When calling `SetViewpointCameraAsync` on a timer with high animation speeds a crash may be encountered.
- Some 3D models do not respect their color, they appear black, on iPhone 5
- The selection halo for a newly added feature is not visible at all scales in SceneView
- Scene layer packages are sometimes visible through the terrain.
- Changing the height of a 3D Scene Symbol changes the anchor position

Raster data

- Raster renderer produces different results between MapView and SceneView
- RasterLayer doesn't free a used raster before the application is shut down.

Copyright © 1995-2020 Esri. All rights reserved.

Geoprocessing

- To return m and z values from the `Submit` geoprocessing tasks, you must include the `returnZ=true` and `return=true` parameters to the call that fetches the output value of the `GeoprocessingFeature`.

Miscellaneous

- Suggestion results from local street address locators cannot be used to retrieve results if they don't contain a house number
- Geodatabase objects may be taken out of scope too soon, so accessing tables once the geodatabase is loaded may fail.
- Locator performance may be poor for larger local locator datasets.
- Calling `pause` on a job that is downloading a file does not pause the job.
- Creating a FeatureCollectionTable from a FeatureQueryResult throws an exception.
- Military symbols requiring geometry conversions where the data is not multipoint are not drawn correctly when you zoom out
- Military symbols for 2525d dictionary which contain labels with non-printable characters may take long time to display on Linux because of font fallback.
- ReturnGeometry set to false is not honored when QueryParameter is passed to PopulateFromServiceAsync
  Arcade expressions using date formatting string will get different results compared to the Javascript webclient.

Related topics

Release notes
Release notes for 100.7.0
Release notes for 100.6.0
Release notes for 100.5.0
Release notes for 100.4.0
Release notes for 100.3.0
Release notes for 100.2.0
Release notes for 100.1.0

Copyright © 1995-2020 Esri. All rights reserved.

System requirements for 100.8.0

This page lists system requirements for the 100.8.0 release of ArcGIS Runtime SDK for .NET, which contains APIs to develop apps for Windows Presentation Foundation (WPF), Universal Windows Platform (UWP), Xamarin.Android, Xamarin.iOS, and Xamarin.Forms (cross-platform projects for Android, iOS and UWP). Each API has specific system requirements for your development machine as well as for the machines and devices to which you deploy your app.

System requirements for recent previous releases are linked to from the bottom of this page. For older system requirements, see past guide versions in PDF format on the downloads page (requires sign-in).

Supported operating systems and configurations for development

The following table gives an overview of supported development configurations for each ArcGIS Runtime SDK for .NET platform.

	Windows 8.1	**Windows 10**	**macOS**
Development Environment	Visual Studio	Visual Studio	Visual Studio for Mac
Windows Presentation Foundation (WPF)	**Yes**	**Yes**	No
Universal Windows Platform (UWP)	No	**Yes**	No
Xamarin.Android	**Yes**	**Yes**	**Yes**
Xamarin.iOS	* **Yes**	* **Yes**	**Yes**
Xamarin.Forms	Android, * iOS	Android, * iOS, UWP	Android, iOS

* Building Xamarin.iOS projects requires an associated Mac computer to act as a build host.

The sections below provide requirements for using ArcGIS Runtime SDK for .NET (Xamarin.iOS) for development on different platforms. For Microsoft's information on setting up your development environment for Xamarin.iOS or Xamarin.Forms see the following topics:

- Prerequisites for using Xamarin
- Installing Xamarin.iOS
- Requirements for Xamarin.Forms
- Current Xamarin Releases

Developing apps on macOS

- A Mac computer running an operating system listed below:
  - macOS Catalina 10.15 minimum
- Development environment: Visual Studio 2019 for Mac
  - Xamarin.iOS — Current Stable channel release recommended

Developing apps on Windows

Supported operating systems:

- Windows 10 Version 1803 minimum
- Windows 8.1
- Development environment:
  - Visual Studio 2019 16.4 minimum (with workload **Mobile development with .NET**)

Copyright © 1995-2020 Esri. All rights reserved.

- Xamarin.iOS - Current Stable channel release recommended

When developing on Windows, you must configure a Mac computer to act as a **build host** for remote compilation and debugging. See the Xamarin installation guide for information about configuring and connecting your Windows development machine to a Mac build host.

Supported emulators/simulators

- iOS Simulator

Deploying your apps

iOS: Your apps can be deployed to devices running any version of iOS 13 or iOS 12.

ArcGIS software and connected services

Use this table to determine the lowest **Version** of the ArcGIS Software that supports a specific ArcGIS Runtime function.

ArcGIS Software	Version	Functionality supported by the ArcGIS Runtime SDKs
ArcGIS Enterprise	10.8	• Standard and advanced Named User licensing supported
	10.7.1	• Scheduled updates to the feature services involved in the preplanned workflow (offline maps)
	10.7	• Annotation layers
	10.6.1	• Offline maps created using the preplanned workflow
	10.4	• Advanced parameters for the on-demand workflow (offline maps)
	10.3	• Web map and scene support. • Feature services. • Tiled services. • Offline maps created using the on-demand workflow.
ArcGIS Pro	2.6	• Online services for layers and tables are supported in mobile map and scene packages.
	2.5	• Elevation sources can use raster files in a mobile scene package. • Transportation networks in a mobile scene package. • Vector tile packages with custom styles in mobile map packages.
	2.4	• Expiration of mobile map and scene packages (requires the Publisher Extension) • Utility network (version 3)

Copyright © 1995-2020 Esri. All rights reserved.

ArcGIS Software	Version	Functionality supported by the ArcGIS Runtime SDKs
	2.3.2	• Create a mobile scene package (.mspk)
	2.3	• Create annotation layers and publish in a mobile map package • Next generation (.loz) locators
	2.2	• Create a mobile map package (.mmpk) • Utility network (version 2)
ArcMap	10.3	• Create Runtime Content

Local Server is supported on Windows when using the WPF API. Local Server is not supported on Android or iOS.

Versions of supported specifications

Specifications supported in this release include the following:

- Augmented reality:
  - ARCore version 1.8 is supported
  - ARKit version 1 is supported
- GeoPackage versions 1.0, 1.1, and 1.2
- Military standards APP-6(B), APP-6(D), MIL-STD-2525B (change 2), MIL-STD-2525C, and MIL-STD-2525D
- Web Feature Service (WFS) versions 2.0.0 and 2.0.2
- Web Map Service (WMS) versions 1.1.0, 1.1.1, and 1.3.0
- WMTS version 1.0.0
- I3S version 1.6

Related topics

Install the SDK
Release notes
Develop your first map app
System requirements for 100.7.0
System requirements for 100.6.0
System requirements for 100.5.0
System requirements for 100.4.0
System requirements for 100.3.0

Copyright © 1995-2020 Esri. All rights reserved.

System requirements for 100.7.0

This page lists system requirements for the 100.7.0 release of ArcGIS Runtime SDK for .NET, which contains APIs to develop apps for Windows Presentation Foundation (WPF), Universal Windows Platform (UWP), Xamarin.Android, Xamarin.iOS, and Xamarin.Forms (cross-platform projects for Android, iOS and UWP). Each API has specific system requirements for your development machine as well as for the machines and devices to which you deploy your app.

System requirements for recent previous releases are linked to from the bottom of this page. For older system requirements, see past guide versions in PDF format on the downloads page (requires sign-in).

Supported operating systems and configurations for development

The following table gives an overview of supported development configurations for each ArcGIS Runtime SDK for .NET platform.

	Windows 7 SP1	**Windows 8.1**	**Windows 10**	**macOS**
Development Environment	Visual Studio	Visual Studio	Visual Studio	Visual Studio for Mac
Windows Presentation Foundation (WPF)	**Yes**	**Yes**	**Yes**	No
Universal Windows Platform (UWP)	No	No	**Yes**	No
Xamarin.Android	**Yes**	**Yes**	**Yes**	**Yes**
Xamarin.iOS	* **Yes**	* **Yes**	* **Yes**	**Yes**
Xamarin.Forms	No	Android, * iOS	Android, * iOS, UWP	Android, iOS

* Building Xamarin.iOS projects requires an associated Mac computer to act as a build host.

The sections below provide requirements for using ArcGIS Runtime SDK for .NET (Xamarin.iOS) for development on different platforms. For Microsoft's information on setting up your development environment for Xamarin.iOS or Xamarin.Forms see the following topics:

- Prerequisites for using Xamarin
- Installing Xamarin.iOS
- Requirements for Xamarin.Forms
- Current Xamarin Releases

Developing apps on macOS

- A Mac computer running an operating system listed below:
  - Catalina 10.15 minimum
- Development environment: Visual Studio 2019 for Mac
  - Xamarin.iOS—Current Stable channel release recommended

Developing apps on Windows

Supported operating systems:

- Windows 10 Version 1709 (Build 16299 / Fall Creators Update)
- Windows 8.1
- Windows 7 SP1

Copyright © 1995-2020 Esri. All rights reserved.

Development environment on Windows	Version
Visual Studio	2019 16.3 minimum (with workload 'Mobile development with .NET')
Simulator: iOS Simulator	

When developing on Windows, you must configure a Mac computer to act as a **build host** for remote compilation and debugging. See the Xamarin installation guide for information about configuring and connecting your Windows development machine to a Mac build host.

Supported emulators/simulators

- iOS Simulator

Deploying your apps

Applications built with this release of ArcGIS Runtime SDK for iOS can be deployed to iOS devices running any version of **iOS 13**, **iOS 12** or **iOS 11**.

Note:
Support for iOS 11 is deprecated. A minimum of iOS 12 will be required at a future release.

ArcGIS software and connected services

Use this table to determine the lowest **Version** of the ArcGIS Software that supports a specific ArcGIS Runtime function.

ArcGIS Software	Version	Functionality supported by the ArcGIS Runtime SDKs
ArcGIS Enterprise	10.8	• Standard and advanced Named User licensing supported
	10.7.1	• Scheduled updates to the feature services involved in the preplanned workflow (offline maps)
	10.7	• Annotation layers
	10.6.1	• Offline maps created using the preplanned workflow
	10.4	• Advanced parameters for the on-demand workflow (offline maps)
	10.3	• Web map and scene support • Feature services • Tiled services • Offline maps created using the on-demand workflow

Copyright © 1995-2020 Esri. All rights reserved.

ArcGIS Software	Version	Functionality supported by the ArcGIS Runtime SDKs
ArcGIS Pro	2.5	• Elevation sources can use raster files in a mobile scene package • Transportation networks in a mobile scene package • Vector tile packages with custom styles in mobile map packages
	2.4	• Expiration of mobile map and scene packages (requires the Publisher Extension) • Utility network (version 3)
	2.3.2	• Create a mobile scene package (.mspk)
	2.3	• Create annotation layers and publish in a mobile map package • Next generation (.loz) locators (only supported with 64-bit apps)
	2.2	• Create a mobile map package (.mmpk) • Utility network (version 2)
ArcMap	10.3	• Create Runtime Content

Local Server is supported on Windows when using WPF. Local Server is not supported on Android or iOS.

Versions of supported specifications

Specifications supported at 100.7.0 include the following:

- Augmented reality:
  - ARCore version 1.8 is supported
  - ARKit version 1 is supported
- GeoPackage versions 1.0, 1.1, and 1.2
- Military standards APP-6(B), APP-6(D), MIL-STD-2525B (change 2), MIL-STD-2525C, and MIL-STD-2525D
- Web Feature Service (WFS) versions 2.0.0 and 2.0.2
- Web Map Service (WMS) versions 1.1.0, 1.1.1, and 1.3.0
- WMTS version 1.0.0
- I3S version 1.6

Related topics

System requirements
System requirements for 100.6.0
System requirements for 100.5.0
System requirements for 100.4.0
System requirements for 100.3.0

Copyright © 1995-2020 Esri. All rights reserved.

System requirements for 100.6.0

This page lists system requirements for the 100.6.0 release of ArcGIS Runtime SDK for .NET, which contains APIs to develop apps for Windows Presentation Foundation (WPF), Universal Windows Platform (UWP), Xamarin.Android, Xamarin.iOS, and Xamarin.Forms (cross-platform projects for Android, iOS and UWP). Each API has specific system requirements for your development machine as well as for the machines and devices to which you deploy your app.

View system requirements for past releases by downloading past guide versions in PDF format from the downloads page (requires sign-in).

Supported operating systems and configurations for development

The following table gives an overview of supported development configurations for each ArcGIS Runtime SDK for .NET platform.

	Windows 7 SP1	**Windows 8.1**	**Windows 10**	**macOS**
Development Environment	Visual Studio	Visual Studio	Visual Studio	Visual Studio for Mac
Windows Presentation Foundation (WPF)	**Yes**	**Yes**	**Yes**	No
Universal Windows Platform (UWP)	No	No	**Yes**	No
Xamarin.Android	**Yes**	**Yes**	**Yes**	**Yes**
Xamarin.iOS	* **Yes**	* **Yes**	* **Yes**	**Yes**
Xamarin.Forms	No	Android, * iOS	Android, * iOS, UWP	Android, iOS

* Building Xamarin.iOS projects requires an associated Mac computer to act as a build host.

The sections below provide requirements for using ArcGIS Runtime SDK for .NET (Xamarin.iOS) for development on different platforms. For Microsoft's information on setting up your development environment for Xamarin.iOS or Xamarin.Forms see the following topics:

- Prerequisites for using Xamarin
- Installing Xamarin.iOS
- Requirements for Xamarin.Forms
- Current Xamarin Releases

Requirements for developing on macOS

- A Mac computer running an operating system listed below:
  - High Sierra 10.13 minimum
- Development environment: Visual Studio 2019 for Mac or Visual Studio 2017 for Mac
  - Xamarin.iOS—Current Stable channel release recommended

Requirements for developing on Windows

Supported operating systems:

- Windows 10 Version 1703 (Build 15063 / Creators Update)
- Windows 8.1
- Windows 7 SP1

Copyright © 1995-2020 Esri. All rights reserved.

Development environment on Windows	Version
Visual Studio 2019	2019 with workload 'Mobile development with .NET'
Visual Studio 2017	2017 with workload 'Mobile development with .NET'
Simulator: iOS Simulator	

When developing on Windows, you must configure a Mac computer to act as a **build host** for remote compilation and debugging. See the Xamarin installation guide for information about configuring and connecting your Windows development machine to a Mac build host.

Supported emulators/simulators

- iOS Simulator

Requirements for deploying your apps

Applications built with this release of ArcGIS Runtime SDK for iOS can be deployed to iOS devices running any version of **iOS 13**, **iOS 12** or **iOS 11**.

Requirements for ArcGIS software and connected services

To use ArcGIS Server data or services in your app, use ArcGIS Server version 10.3 or later. To use ArcGIS Server data when your app is offline, use ArcGIS Server version 10.3 or later.

To create mobile map packages for use in your ArcGIS Runtime SDK apps, use ArcGIS Pro version 2.2 or later.

To create mobile scene packages for use in your ArcGIS Runtime SDK apps, use ArcGIS Pro version 2.3 or later.

To create next generation locators (`*.loz`), use ArcGIS Pro version 2.3 or later. Next generation locators are only supported on 64-bit apps.

To create annotation layers for use in your ArcGIS Runtime SDK apps, use ArcGIS Pro version 2.3 or later. Annotation layers can be published to mobile map packages or to a service that uses ArcGIS Enterprise version 10.7 or later.

ArcGIS Desktop version 10.3 or later is required if you're using the Create Runtime Content capability. You can create ArcGIS Runtime SDK content in the ArcMap user interface or with the Create Runtime Content geoprocessing tool.

ArcGIS Runtime SDK works with ArcGIS Enterprise version 10.3 or later portals, and web maps created with these portals.

ArcGIS Runtime SDK works with utility network schema versions 2 and 3.

Local Server is supported on Windows. Local Server is not supported on Android or iOS.

Versions of supported specifications

Specifications supported at 100.6.0 include the following:

- Augmented reality:
  - ARCore version 1.8 is supported
  - ARKit version 1 is supported
- GeoPackage versions 1.0, 1.1, and 1.2
- Military standards APP-6(B), APP-6(D), MIL-STD-2525B (change 2), MIL-STD-2525C, and MIL-STD-2525D
- Web Feature Service (WFS) versions 2.0.0 and 2.0.2
- Web Map Service (WMS) versions 1.1.0, 1.1.1, and 1.3.0

Copyright © 1995-2020 Esri. All rights reserved.

- WMTS version 1.0.0
- I3S version 1.6

Related topics

System requirements
System requirements for 100.7.0
System requirements for 100.5.0
System requirements for 100.4.0
System requirements for 100.3.0

Copyright © 1995-2020 Esri. All rights reserved.

System requirements for 100.5.0

ArcGIS Runtime SDK for .NET contains APIs to develop apps for Windows Presentation Foundation (WPF), Universal Windows Platform (UWP), Xamarin.Android, Xamarin.iOS and Xamarin.Forms (cross-platform projects for Android, iOS, and UWP). Each API has specific system requirements for your development machine as well as for the machines and devices to which you will deploy your app.

Note:
You can view system requirements for past SDK releases by downloading past guide versions in PDF format from the downloads page (requires sign-in).

The following table gives an overview of supported development configurations for each ArcGIS Runtime SDK for .NET platform.

	Windows 7 SP1	Windows 8.1	Windows 10	macOS
Development Environment	Visual Studio	Visual Studio	Visual Studio	Visual Studio for Mac
Windows Presentation Foundation (WPF)	Yes	Yes	Yes	No
Universal Windows Platform (UWP)	No	No	Yes	No
Xamarin.Android	Yes	Yes	Yes	Yes
Xamarin.iOS	* Yes	* Yes	* Yes	Yes
Xamarin.Forms	No	Android, * iOS	Android, * iOS, UWP	Android, iOS

* Building Xamarin.iOS projects requires an associated Mac computer to act as a build host.

Xamarin.iOS

For detailed information on setting up your development environment for Xamarin.iOS or Xamarin.Forms, see the following topics:

- Prerequisites for using Xamarin
- Installing Xamarin.iOS
- Requirements for Xamarin.Forms
- Current Xamarin Releases

Requirements for development

On Windows

Note:
When developing on Windows, you need to configure a Mac computer to act as a build host for remote compilation and debugging. See the Xamarin installation guide for information about configuring and connecting your Windows development machine to a Mac build host.

Operating system

- Windows 10

Copyright © 1995-2020 Esri. All rights reserved.

- Windows 8.1
- Windows 7 SP1

Development Environment

- Visual Studio 2017 (with workload 'Mobile development with .NET')
- Visual Studio 2019 (with workload 'Mobile development with .NET')

On Mac

Operating system

- macOS High Sierra (10.13) minimum.

Development environment

- Visual Studio 2017 for Mac
- Visual Studio 2019 for Mac
- Xamarin.iOS (current Stable channel release recommended)

Simulator

- iOS Simulator

Requirements for deployment

iOS 11.0 minimum. Latest version is recommended.

Requirements for ArcGIS software and connected services

To use ArcGIS Server data or services in your app, use ArcGIS Server version 10.2 (or higher). To use ArcGIS Server data when your app is offline, use ArcGIS Server version 10.2.2 (or higher).

To create mobile map packages for use in your ArcGIS Runtime SDK apps, use ArcGIS Pro version 1.3 (or higher).

To create mobile scene packages for use in your ArcGIS Runtime SDK apps, use ArcGIS Pro version 2.3 (or higher).

To create annotation layers for use in your ArcGIS Runtime SDK apps, use ArcGIS Pro version 2.3 (or higher). Annotation layers can be published to mobile map packages or to a service that uses ArcGIS Enterprise version 10.7 (or higher).

ArcGIS Desktop version 10.3 (or higher) is required if you're using the Create Runtime Content capability. You can create ArcGIS Runtime SDK content in ArcMap's user interface or with the Create Runtime Content geoprocessing tool.

ArcGIS Runtime SDK works with ArcGIS Enterprise version 10.2 (or higher) portals. However, when your app uses web maps, it must use web maps built with the 2.0 (or higher) version of the web map specification, which come from Enterprise version 10.2 (or higher) portals. When you save a web map from an app, you are saving a web map that adheres to version 2.9 of the web map specification. These should only be saved back to portals that support the 2.0 web map specification.

Local Server is supported on Windows. Local Server is not supported on Android or iOS.

Versions of supported specifications

Specifications supported at 100.5.0 include the following.

- GeoPackage versions 1.0, 1.1, and 1.2

Copyright © 1995-2020 Esri. All rights reserved.

- Military standards:
  - APP-6(B)
  - APP-6(D)
  - MIL-STD-2525B (change 2)
  - MIL-STD-2525C
  - MIL-STD-2525D
- Web Feature Service (WFS) versions 2.0.0 and 2.0.2
- Web Map Service (WMS) versions 1.1.0, 1.1.1, and 1.3.0
- WMTS version 1.0.0
- I3S version 1.6

Related topics

System requirements
System requirements for 100.7.0
System requirements for 100.6.0
System requirements for 100.4.0
System requirements for 100.3.0

Copyright © 1995-2020 Esri. All rights reserved.

System requirements for 100.4.0

ArcGIS Runtime SDK for .NET contains APIs to develop apps for Windows Presentation Foundation (WPF), Universal Windows Platform (UWP), Xamarin.Android, Xamarin.iOS and Xamarin.Forms (cross-platform projects for Android, iOS and UWP). Each API has specific system requirements for your development machine as well as for the machines and devices to which you will deploy your app.

Note:
You can view system requirements for past SDK releases by downloading past guide versions in PDF format from the downloads page (requires sign-in).

The following table gives an overview of supported development configurations for each ArcGIS Runtime SDK for .NET platform.

	Windows 7 SP1	Windows 8.1	Windows 10	macOS
Development Environment	Visual Studio	Visual Studio	Visual Studio	Visual Studio for Mac
Windows Presentation Foundation (WPF)	**Yes**	**Yes**	**Yes**	No
Universal Windows Platform (UWP)	No	No	**Yes**	No
Xamarin.Android	**Yes**	**Yes**	**Yes**	**Yes**
Xamarin.iOS	* **Yes**	* **Yes**	* **Yes**	**Yes**
Xamarin.Forms	No	Android, * iOS	Android, * iOS, UWP	Android, iOS

* Building Xamarin.iOS projects requires an associated Mac computer to act as a build host.

Xamarin.iOS

For detailed information on setting up your development environment for Xamarin.iOS or Xamarin.Forms see the following topics:

- Prerequisites for using Xamarin
- Installing Xamarin.iOS
- Requirements for Xamarin.Forms
- Current Xamarin Releases

Requirements for development

On Windows

Note:
When developing on Windows, you will need to configure a Mac computer to act as a **build host** for remote compilation and debugging. See the Xamarin installation guide for information about configuring and connecting your Windows development machine to a Mac build host.

Operating System

- Windows 10

Copyright © 1995-2020 Esri. All rights reserved.

- Windows 8.1
- Windows 7 SP1

Development Environment

- Visual Studio 2017 (with workload 'Mobile development with .NET')

On Mac

Operating System

- macOS High Sierra (10.13) minimum.

Development Environment

- Visual Studio 2017 for Mac
- Xamarin.iOS (current Stable channel release recommended)

Simulator

- iOS Simulator

Requirements for deployment

iOS 10.0 minimum. Latest version is recommended.

Requirements for ArcGIS software and connected services

To use ArcGIS Server data or services in your app, use ArcGIS Server version 10.2 (or higher). To use ArcGIS Server data when your app is offline, use ArcGIS Server version 10.2.2 (or higher).

To create mobile map packages for use in your ArcGIS Runtime SDK apps, use ArcGIS Pro version 1.3 (or higher).

ArcGIS Desktop version 10.3 (or higher) is required if you're using the Create Runtime Content capability. You can create ArcGIS Runtime SDK content in ArcMap's user interface or with the Create Runtime Content geoprocessing tool.

ArcGIS Runtime SDK works with ArcGIS Enterprise version 10.2 (or higher) portals. However, when your app uses web maps, it must use web maps built with the 2.0 (or higher) version of the web map specification, which come from Enterprise version 10.2 (or higher) portals. When you save a web map from an app, you are saving a web map that adheres to version 2.9 of the web map specification. These should only be saved back to portals that support the 2.0 web map specification.

Local Server is supported on Windows. Local Server is not supported on Android or iOS.

Versions of supported specifications

Specifications supported at 100.8.0 include the following.

- GeoPackage versions 1.0, 1.1, and 1.2
- Military standards:
  - APP-6(B)
  - APP-6(D)
  - MIL-STD-2525B (change 2)
  - MIL-STD-2525C

Copyright © 1995-2020 Esri. All rights reserved.

  - MIL-STD-2525D
- Web Map Service (WMS) versions 1.1.0, 1.1.1, and 1.3.0
- WMTS version 1.0.0

On-premises virtualization

On-premises virtual desktop infrastructure (VDI)

Product	Notes
VMware Horizon View 6.5 VMware Horizon View 7.0	Hypervisor: VMware vSphere 6.5 Use the latest VMware and NVIDIA drivers. Refer to VMware and NVIDIA GRID websites for the most current driver versions.
Citrix XenDesktop 7.17 Citrix XenDesktop 7.18	Hypervisor: VMware vSphere 6.5 Use the latest Citrix and NVIDIA drivers. For the most current driver versions, see the Citrix and NVIDIA GRID websites.

Related topics

System requirements
System requirements for 100.7.0
System requirements for 100.6.0
System requirements for 100.5.0
System requirements for 100.3.0

Copyright © 1995-2020 Esri. All rights reserved.

System requirements for 100.3.0

ArcGIS Runtime SDK for .NET contains APIs to develop apps for Windows Presentation Foundation (WPF), Universal Windows Platform (UWP), Xamarin.Android, Xamarin.iOS and Xamarin.Forms (cross-platform projects for Android, iOS and UWP). Each API has specific system requirements for your development machine as well as for the machines and devices to which you will deploy your app.

The following table gives an overview of supported development configurations for each ArcGIS Runtime SDK for .NET platform.

	Windows 7 SP1	**Windows 8.1**	**Windows 10**	**macOS**
Development Environment	Visual Studio	Visual Studio	Visual Studio	Visual Studio for Mac
Windows Presentation Foundation (WPF)	**Yes**	**Yes**	**Yes**	No
Universal Windows Platform (UWP)	No	No	**Yes**	No
Xamarin.Android	**Yes**	**Yes**	**Yes**	**Yes**
Xamarin.iOS	* **Yes**	* **Yes**	* **Yes**	**Yes**
Xamarin.Forms	No	Android, * iOS	Android, * iOS, UWP	Android, iOS

* Building Xamarin.iOS projects requires an associated Mac computer to act as a build host.

Xamarin.iOS

For detailed information on setting up your development environment for Xamarin.iOS or Xamarin.Forms see the following topics:

- Prerequisites for using Xamarin
- Installing Xamarin.iOS
- Requirements for Xamarin.Forms
- Current Xamarin Releases

Requirements for development

On Windows

Note:
When developing on Windows, you will need to configure a Mac computer to act as a **build host** for remote compilation and debugging. See the Xamarin installation guide for information about configuring and connecting your Windows development machine to a Mac build host.

Supported operating systems:

- Windows 10
- Windows 8.1
- Windows 7 SP1

Supported IDEs:

Copyright © 1995-2020 Esri. All rights reserved.

- Visual Studio 2017
- Visual Studio 2015 Update 3

Xamarin products:

- Xamarin for Visual Studio (current Stable channel release recommended)
- Xamarin.iOS (current Stable channel release recommended)

On Mac

Supported operating systems:

- macOS High Sierra (10.13)

Supported IDEs:

- Visual Studio 2017 for Mac

Xamarin products:

- Xamarin.iOS (current Stable channel release recommended)

Supported simulators

- iOS Simulator

Requirements for deployment

iOS 10.0 minimum. Latest version is recommended.

Requirements for ArcGIS software and connected services

To use ArcGIS Server data or services in your app, use ArcGIS Server version 10.2 (or higher). To use ArcGIS Server data when your app is offline, use ArcGIS Server version 10.2.2 (or higher).

To create mobile map packages for use in your ArcGIS Runtime SDK apps, use ArcGIS Pro version 1.3 (or higher).

ArcGIS Desktop version 10.3 (or higher) is required if you're using the Create Runtime Content capability. You can create ArcGIS Runtime SDK content in ArcMap's user interface or with the Create Runtime Content geoprocessing tool.

ArcGIS Runtime SDK works with ArcGIS Enterprise version 10.2 (or higher) portals. However, when your app uses web maps, it must use web maps built with the 2.0 (or higher) version of the web map specification, which come from Enterprise version 10.2 (or higher) portals. When you save a web map from an app, you are saving a web map that adheres to version 2.9 of the web map specification. These should only be saved back to portals that support the 2.0 web map specification.

Local Server is supported on Windows. Local Server is not supported on Android or iOS.

Versions of supported specifications

Specifications supported at 100.3.0 include the following.

- GeoPackage versions 1.0, 1.1, and 1.2
- Military standards: APP-6(B), APP-6(D), MIL-STD-2525B (change 2), MIL-STD-2525C, and MIL-STD-2525D
- Web Map Service (WMS) versions 1.1.0, 1.1.1, and 1.3.0
- WMTS version 1.0.0

Copyright © 1995-2020 Esri. All rights reserved.

Related topics

System requirements
System requirements for 100.7.0
System requirements for 100.6.0
System requirements for 100.5.0
System requirements for 100.4.0

Copyright © 1995-2020 Esri. All rights reserved.

Essential vocabulary

Term	Definition
advanced symbol	See multilayer symbol.
advanced symbology	The set of conventions, rules, or encoding systems that define how to represent on a map geographic features or other items of interest (such as airplane routes) that are more complex than a simple point, line, or polygon. The advanced symbology category includes the following: • Geodesics • Features symbolized with symbol dictionaries, such as features that adhere to military specifications (for example, warfighting symbols defined in MIL-STD-2525C)
analysis	A systematic examination of a problem that provides new information. ArcGIS Runtime supports many types of analyses, from simple geometry-based analysis to advanced spatial analysis. You can also string analysis operations together to build models. Stringing operations together for modeling or for the automation of repetitive workflows (for example, batch processing) is known as geoprocessing.
annotation layer	Represents a layer used to visualize annotations, similar to the way a feature layer visualizes features. An annotation is a type of feature that consists of text with position, layout and, style. Currently, ArcGIS Runtime supports read-only annotation.
API	Application programming interface. API is a specification that allows developers to create applications.
ARCore	A Google SDK that enables the development of augmented reality (AR) experiences. It uses a device's camera to understand the environment and provide precise and accurate position information within a range of a few meters from the origin.
ArcGIS Developer Program	ArcGIS Developer Program (ADP) is a developer-focused online community that provides users with the best possible experience for discovering the development and business opportunities that ArcGIS provides. ArcGIS Developer Subscription plans allow developers to use various ArcGIS capabilities by choosing a subscription plan that best aligns with their development skills and business goals.
ArcGIS Marketplace	ArcGIS Marketplace is an online marketplace for customers to discover valuable new capabilities and applications, and connect directly with developers to obtain new solutions.
ArcGIS platform	Server technology (that can be hosted by you or Esri), data, apps, APIs, and other elements designed to work together to provide anything from small-scale, focused GIS solutions (such as an app that allows you to collect data in the field and sync it to your database) to complex, enterprise-level solutions that model real-world scenarios and use analytical capabilities to solve complex problems.
ARKit	An Apple SDK that enables the development of augmented reality (AR) experiences. It uses a device's camera to understand the environment and provide precise and accurate position information within a range of a few meters from the origin.
asset group (`UtilityAssetGroup`)	The first-level categorization of a utility element. Each asset group has a collection of asset types that provide the second-level categorization. An example for an electric utility is the asset group High Voltage Insulator, which contains the asset types Single, Single String Running Angle, Double Suspension, and Triple String Running Angle. Also known as subtype.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
asset type (`UtilityAssetType`)	A class that defines a utility element's classification within an asset group. For example, a power transformer is an asset type of a transformer asset group in an electric device network. A terminal configuration can be assigned to one or more asset types. For more information, see ArcGIS Pro's help topic about utility feature classifications.
association (`UtilityAssociation`)	A relationship between two elements that is reflected in the network topology. For more information, see the discussion on how to get associated utility elements in this guide. Some tracing methods make use of associations and can even return the associated structures or containers.
association graphic	A graphic that gives a visual representation of a structural or attachment association between two utility network elements.
asynchronous	Within the context of discussing how ArcGIS APIs work, asynchronous is a concept by which code is executed on an available background thread in the application's thread pool. The results of this executed code (if any) are returned upon completion. This allows you to off-load execution of longer-running processes, freeing up the UI thread so the app remains responsive to user interaction.
attachment	A file, such as a photograph (for example, a `.png` file) or a document, that's associated with a feature in the geodatabase.
augmented reality (AR)	A type of development pattern in ArcGIS Runtime that allows you to provide your users with an immersive 3D experience using the movement of their device rather than manual touch or mouse-based interactions.
Authentication Manager	A prebuilt component you can add to your app that displays a dialog box asking for user credentials whenever an attempt is made on a secure resource (such as a secure layer) where the credentials are missing or invalid.
base layer	An element of a basemap. A basemap can contain a number of base layers that can give your map a recognizable background so that your operational layers are displayed in context. Base layers are always at the bottom of a map.
basemap	A map depicting background reference information such as landforms, roads, landmarks, and political boundaries, onto which other thematic information is placed. For a description of basemaps and other layer types, see Layers and tables.
billboarding	A way to display symbols in 3D (in scenes) by posting them vertically as 2D symbols and orienting them to always face the camera.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
by reference basemap	A map that is going to be taken offline can use a basemap that already exists on the device. The use of a local basemap is called by reference basemap. It is identified by a reference basemap file name and a directory property that you can set using the on-demand parameters (see Create parameters to specify map content or the planned parameters (see Take a map offline - preplanned.
cache	See tile cache.
callout	A container view that can be added over the map and anchored to a coordinate with a leader. The callout leader or tail indicates the location to which it's referring and has a configurable area that can contain other GUI and View components. These can be anything you add to the callout, including a title, an image, text, and a pop-up. A map can have one callout. Also see pop-up.
camera	A camera in a 3D scene is a virtual device that provides the rendering viewpoint. A camera has a location (x longitude, y latitude, and z elevation), heading (angle about the x-axis the camera is rotated, in degrees), pitch (angle the camera is rotated relative to the y-axis, in degrees), and roll (angle the camera is rotated about its center axis, in degrees.)
canvas	A kind of ArcGIS Online basemap specifically designed to give users a neutral canvas on which to better display data.
canvas layer	A basemap layer that provides a neutral background with minimal colors, labels, and features. Only key information is represented in a canvas layer to provide geographic context, allowing your thematic data to come to the foreground.
child layer	A layer that is a member of a group layer.
CIM symbol	Cartographic Information Model symbols, which are advanced, multilayer symbols produced by ArcGIS Pro and ArcGIS Desktop.
Client ID	In ArcGIS Runtime SDKs, an identifier you associate with the app you build (one Client ID per app). You get a Client ID by signing in to developers.arcgis.com.
cloud	A network of remote servers hosted on the internet to store, manage, and process data. Esri's cloud offering is ArcGIS Online, a collaborative content management system for maps, apps, data, and other geographic content. For details about using ArcGIS Online with your ArcGIS Runtime app, see Access the ArcGIS platform.
composite relationship	A relationship between two tables in which each feature in the destination table is expected to be related to an origin feature. In other words, any orphan destination features are considered a violation of the relationship. For example, a building and its mailbox must be related. While the building (origin) can exist on its own, a mailbox (destination) must be related to a building. When an origin feature is deleted, the destination feature should also be deleted. This is known as a cascade delete. Also see simple relationship.
composite symbol	A symbol that is a combination of two or more symbols. Each symbol can represent the same or a different aspect of a graphic or feature.
configurable app	A software program that you can modify, typically focused on a specific task, such as finding directions, that's hosted on ArcGIS Online. Typically, the source code for the app is available on GitHub for you to customize. See the configurable apps gallery on ArcGIS Online.
connected	1. In ArcGIS Runtime SDKs, the state of having a network connection to ArcGIS Online or ArcGIS Enterprise. 2. In ArcGIS, a type of utility network trace. See connected trace.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
connected trace	A trace that begins at one or more starting points and spans outward radially along connected features. A trace stops when a barrier is encountered or there are no more connected features. This type of trace is useful for validating newly edited features to ensure they are connected as expected. For details, see this guide's topic on performing traces or see ArcGIS Pro's help topics on connectivity and finding connected features.
coordinate	A value that denotes the location of a vertex. Coordinates can represent 2D (x,y) or 3D (x,y,z) space. The meaning of the x,y,z coordinates is determined by a coordinate system. The vertices and coordinate system together allow your app to translate a real-world object from its location on the earth to its location on your map. For details, see Geometries.
coordinate system	A reference framework consisting of a set of points, lines, or surfaces, and a set of rules, used to define the positions of points in space in two or three dimensions. For details, see Geometries. This is also known as map projections. Also see projection.
cross-platform development	The capability supported by some ArcGIS Runtime SDKs that allows you to program with an SDK on multiple platforms. For example, ArcGIS Runtime SDK for Java allows you to develop on Windows, Linux, and Mac.
cross-platform deployment	The capability supported by several ArcGIS Runtime SDKs that allows you to write once and deploy to multiple platforms and devices. For example, ArcGIS Runtime SDK for Qt allows you to write once and deploy to Android, iOS, Windows, and Linux.
destination feature	A feature in a destination table that is associated with an origin feature in an origin table. You create associations such as this for a variety of reasons, such as to view information in one table for features in another. For more information, see Relate features in this ArcGIS Runtime guide or Essentials of relating tables.
destination table	A table that has been associated with another table, an origin table, using a key. You associate two tables for a variety of reasons, such as to view information in one table for features in another. For more information, see Relate features in this ArcGIS Runtime guide or Essentials of relating tables.
device	In ArcGIS Runtime, nearly any kind of computer, including desktops, laptops, mobile phones, and tablets.
dictionary renderer	A renderer that uses a style file generated in ArcGIS Pro together with a rule engine to display some types of advanced symbology on a map, such as military symbology.
direct connect	A type of connection from ArcGIS directly to a geodatabase instead of accessing the geodatabase through a service.
dirty area	A sublayer of a utility network that captures features that have been edited since the last topology network update/validation. After the topology network is updated, the edits (dirty areas) are incorporated and your utility network should have no more dirty areas. A dirty area is also an edit that violates restrictions for a utility network.
disconnected	The state of not having a network connection to ArcGIS Online or ArcGIS Enterprise. For details on ArcGIS Runtime's offline capabilities, see Offline.
distance composite scene symbol	A type of Symbol that changes based on the distance, in meters, between the SceneView's Camera and the graphic or feature to which the Symbol is assigned.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
domain network (`UtilityDomainNetwork`)	In ArcGIS, the highest-level category used to organize a utility network. It represents the utility system type, such as electric, gas, and water. Or the utility network may have separate distribution and transmission domain networks. For example, for electricity, you might have one for electric transmission and one for electric distribution. The number of domain networks you use depends on the type of model you're building and the industry or industries you manage. All the domain networks in a utility network share the same structure network so that you can find the devices and lines that are supported by common structures. Each domain network contains a tier along with a subnetwork controller type to indicate whether the subnetwork controllers are acting as sinks or sources. For more information, see the domain network topic in the ArcGIS Pro documentation.
downstream trace	In a source-based network (gas or electric), a downstream trace is with the flow and away from the source, such as a circuit breaker or generator (subnetwork controller). In a sink-based network (sewer or storm water), a downstream trace is with the flow and toward the sink such as a sewage treatment (subnetwork controller).For details, see this guide's topic on performing traces or see ArcGIS Pro's help topic on downstream traces.
dynamic labels	Labels that are placed on the fly, and whose text is generated dynamically based on a labeling expression stored with the data being labeled.
dynamic layer	A layer, from a map published through a map service, whose appearance—such as labeling, layer order, and symbology—can be changed by the client.
dynamic map service	A map that is drawn by the server each time the user zooms or pans. This differs from a tiled service in that it does not work with a cache of precooked tiles.
dynamic rendering mode	One of two modes for rendering a graphics overlay. In this mode, which is good in most use cases, the entire graphic representation is stored on the GPU. This mode is especially good for moving objects, as individual graphic changes can be efficiently applied directly to GPU state, resulting in a seamless experience. However, the volume of graphics has a direct impact on (GPU) resources, and the number and complexity of graphics that can be rendered in this mode are dependent on the power of the GPU and the amount of GPU memory. The other rendering mode is called static. For a comparison, see the definition for static rendering mode.
feature	A representation of a real-world object on a map, such as a building, a utility system asset, a river, or a county. A feature is persisted in a feature table in a data store (such as a database or service) or in a map. Features in the same data store or feature layer have a common attribute schema. For more information, see Features and graphics.
feature collection	A data structure containing one or more features having the same set of attributes.
feature data	A general term for the data related to one or more features.
feature layer	A layer that references a set of feature data. Feature data represents geographic entities as points, lines, and polygons.
feature service	A service that streams features. The server bundles feature data and streams it to the requesting client. There are a number of modes that client APIs can use to fetch data from the server and cache features locally if and when necessary.
feature service table	A data structure representing feature data retrieved from a feature service.
feature table	A database table of a single geometry type, such as point, line, or polygon, that stores features that conform to the schema of the table.
field of view (FOV)	The amount of the world that is visible to a camera, usually measured in degrees.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
filter barrier	A setting in the trace configuration that specifies when an isolation trace will stop for a specific category or network attribute. It is required for isolation traces because it pinpoints which features are isolating the starting point or points. For example, stop a trace at features that have a life cycle status attribute that is equal to a certain value. For details, see this guide's topic on performing traces or see ArcGIS Pro's help topic on locating isolating features.
geocode	The process of transforming a description of a location—such as a pair of coordinates, an address, or a name of a place—to a location on the earth's surface. The resulting locations are output as geographic features with attributes, which can be used for mapping or spatial analysis. This is also known as address matching.
geodatabase	A collection of geographic datasets of various types held in a common file system folder, a Microsoft Access database, or a multiuser relational DBMS (such as Oracle, Microsoft SQL Server, PostgreSQL, Informix, or IBM DB2). Geodatabases come in many sizes, have varying numbers of users, and can scale from small, single-user databases built on files up to larger workgroup, department, and enterprise geodatabases accessed by many users.
geodatabase feature	A representation of a real-world object persisted in a geodatabase. When displayed, it's displayed in a feature layer. A feature is associated with a feature table with which it shares a common schema. Common types of feature tables and schemas are point, line, and polygon.
geodesic	The shortest distance between two points on the surface of a spheroid (also known as an ellipsoid). Any two points along a meridian form a geodesic line. This is similar to the great circle method, which models the surface as a sphere.
geodetic measurement	Any method of measuring geographic distance or area that uses a curved (nonplanar) surface to model the earth. The most common of these use a great circle or a geodesic line.
GeoElement	A marker interface for features and graphics that's used when you want to identify (display attribute information on) visible items on a map view.
geometry	The combination of location and shape for a real-world object or a geometric construct such as an area of interest or a buffer area around an object. Geometry is a fundamental element for performing spatial analysis. For more information, see Geometries.
GeoPackage	GeoPackage is an open, standards-based, platform-independent, portable, self-describing, compact format for transferring geospatial information. It uses a single SQLite file (`.gpkg`) that conforms to the OGC GeoPackage specification (http://www.opengeospatial.org/standards/geopackage) to store feature tables, nonspatial tables, raster datasets (images), and metadata.
geoprocessing	A GIS operation used to manipulate data. A typical geoprocessing operation takes an input dataset, performs an operation on that dataset, and returns the result of the operation as an output dataset. Common geoprocessing operations include geographic feature overlay, feature selection and analysis, topology processing, raster processing, and data conversion. Geoprocessing allows for definition, management, and analysis of information used to form decisions. You can use geoprocessing tools to create a sequence of operations, feeding the output of one tool into another tool; automate your work (for example, overnight processing); or solve complex problems using models. Your ArcGIS Runtime app can consume online geoprocessing services.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
geoview	GeoView is a base class for MapView and SceneView. These represent the View in a Model View Controller (MVC) architecture. The Map and Scene that are set on these views represent the model. The GeoView contains all of the common operations and events that apply to displaying and working with Maps and Scenes. This includes changing what is viewable by setting a viewpoint, responding to viewpoint change events, working with graphics overlays and identifying elements that are displayed at a given location in the view.
graphic	A representation of a real-world object stored in memory. When you want to display graphics, you use a graphics overlay. Graphics exist while the app is running and, therefore, are used often for temporary features. Graphics can have geometry and attributes. Graphics are not associated with a feature table. For more information, see Features and graphics.
graphics overlay	An item you use in your map, typically when you have graphics that change location regularly, and you want optimal animation of the graphics when zooming in and out on the map. It differs from a layer because its graphics are temporary (held in device memory) instead of being persisted in the map. For more information, see Features and graphics.
great circle	A geodetic measurement that models the earth's surface as a sphere. A plane is formed by the start and end measure points and the center of the sphere. The intersection of this plane and the spheroid surface represents the shortest distance between the points. This is similar to the geodesic method, which models the surface as a spheroid or an ellipsoid.
group layer	A container for other layers and group layers. It is used to represent datasets that are composed of multiple layers to be managed as a single layer with respect to display and some other operations.
hillshade	A representation of a raster image that portrays a hypothetical illumination of a surface by determining illumination values for each cell in a raster. It can greatly enhance the visualization of a surface for analysis or graphical display, especially when using transparency. By default, shadow and light are shades of gray associated with integers from 0 to 255 (increasing from black to white).
identify	To display, on a map, attribute data of a feature, graphic, or raster cell.
image service	A web service that serves raster and image data. For more information, see the following: • Publish an image service in the ArcGIS Pro documentation • Map image, elevation, and imagery layers in the ArcGIS Online documentation
integrated mesh layer	Layer used to display an integrated mesh in a scene. Typically captured by an automated process for constructing 3D objects out of large sets of overlapping imagery, the result integrates the original input image information into a textured mesh including 3D objects such as buildings and trees, along with elevation information.
isolation trace	A trace used to determine the minimum set of operable assets (point and line features) required to stop a network's resource from traveling/flowing, effectively isolating an area of the network. For instance, when a leak occurs on a water network, particular valves must be closed to eliminate water flow at the leak location. This prevents damage and allows field crews to safely start the repair process. For details, see this guide's topic on performing traces or see ArcGIS Pro's help topic on locating isolating features.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
job	A unit of work performed by a computing system in response to a scheduled or unscheduled request.
KML	Keyhole Markup Language (KML) is an XML-based format for describing geographic entities. Developed and popularized for use with Google Earth, the KML specification is now maintained by the Open Geospatial Consortium (OGC). You can use a `.kml` or `.kmz` (compressed) file, or a URL pointing to a KML file as the source for a layer in your ArcGIS Runtime app.
KML node	An element definition within a KML document. A KML node can be a container (such as a folder that has additional child nodes) or a geometry node (a node that has an associated geometry). Geometry nodes can be placemarks, 3D models, polygons, lines, or multigeometry. A KML node can have nested XML elements and attributes to define things such as display balloons (pop-ups/callouts), ground overlays, screen overlays, folders/documents, network links, and so on.
label	Text displayed with and associated with a graphic or feature.
layer	An item used to display geographic data in a map. In general, a layer's data comes from a single source, such as a map service URL or geodatabase table. A layer uses an associated renderer to symbolize data and in some cases define properties for the display of things such as labels and pop-ups. This decoupling of the layer's data and how it's rendered gives you the flexibility to present the same data in a variety of ways. As the name indicates, layers are stacked, or layered, in the map and drawn from bottom to top. As a developer, you can control the order of the layer in the map as well as its visibility. For more information on layers and how they're used in ArcGIS Runtime, see Layers and tables.
layer definition	A SQL WHERE clause that selects a subset of features from a layer.
legend	A table or chart associated with a map to indicate the meaning of the map's varied symbols and layer representations.
level of detail (LOD)	Scale levels to include in a tiling scheme.
license key	A string of characters you add to your application code to unlock certain capabilities on the deployment device.
loadable	A characteristic of a resource, such as a layer, map, or portal item, that allows you to do such tasks as the following: • Monitor the load status of the resource. • Retry to load if previous load attempts failed. A loadable resource handles concurrent and repeated requests to load to accommodate the common practice of sharing the same resource instance among various parts of an app. For details, see Loadable pattern for asynchronous resources.
Local Server	In ArcGIS Runtime SDKs for Java, .NET (Desktop), and Qt, a miniserver for serving local services that don’t require an internet connection. The miniserver allows you to perform analysis and geoprocessing that's not natively supported in the Runtime core.
locator	A dataset that contains information including address attributes, indexes, and queries for geocoding. An address locator contains a snapshot of the reference data that is used for geocoding. In the process of geocoding, the reference data is no longer needed after the locator is created. A locator can be used to find addresses or x.y locations.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
map	A graphical representation of spatial relationships of entities within an area. In ArcGIS Runtime, a map works together with a map view to provide a visualization of geographic data on a screen. A map specifies how the geographic data is organized; a map view renders the data and allows users to interact with it.
map cache	See tile cache.
map image layer	A layer whose map images are created on the fly as a user zooms and pans around a map. It differs from a tiled layer, whose map images are pregenerated and are displayed as tiles in a layer in the client app. Use a map image layer for content authored in ArcGIS Pro or ArcMap and published as an ArcGIS map service to ArcGIS Online, ArcGIS Enterprise, and Local Server.
map view	A class (`MapView`) that represents the view tier in an MVC architecture. While a map specifies how the geographic data is organized, a map view renders the data and allows users to interact with it.
mobile geodatabase	A geodatabase (.geodatabase) that uses SQLite and can be used in disconnected workflows in ArcGIS Runtime apps.
mobile basemap layer	A compressed data format that can be created from ArcGIS Pro when you share a map as a mobile map package.
mobile map package	A mobile map package (`.mmpk`) allows you to transport and share maps, layers, and data across the ArcGIS platform. You can create mobile map packages using ArcGIS Pro, share them using your ArcGIS organization, or distribute directly by copying to a device, and consume them using ArcGIS Runtime apps. You can also create and download a mobile map package directly to your device using the preplanned or on-demand workflows provided by the Runtime API. Each mobile map package includes metadata about the package along with one or more maps and the data needed to display them. A mobile map package can be delivered as a single (`.mmpk`) file or as a directory structure. See the system requirements for details on what mobile map package versions are currently supported.
mobile mosaic dataset	A SQLite database that allows you to store, manage, view, and query small to vast collections of raster and image data. The mobile mosaic dataset can be used as a raster layer source for mapping in disconnected workflows in ArcGIS Runtime apps. They can also be created and modified using any of the ArcGIS Runtime native APIs. See Add raster data for more information.
mobile scene package	A mobile scene package (`.mspk`) allows you to transport and share scenes, layers, elevation sources, and data across the ArcGIS platform. You can create mobile scene packages using ArcGIS Pro, share them using your ArcGIS organization, or distribute directly by copying to a device, and consume them using ArcGIS Runtime apps. Each mobile scene package includes metadata about the package (description, thumbnail, and so on) along with one or more scenes and the data needed to display them. See the system requirements for details on what mobile scene package versions are currently supported.
model symbol	A type of marker symbol that uses a true three-dimensional graphical model to define the marker. The model used is derived from a URI to a model in a supported 3D model format.
mosaic dataset	A grouping of raster data images into a single container for convenient display. See Mosaic datasets in the ArcGIS Pro documentation for more information.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
multilayer symbol	A symbol type that's often used for symbols that require more than a single label. For example, military symbols require multiple adornments and text to display in predefined positions around a base symbol. Multilayer symbols are often used for military symbols, because each layer in the symbol can contain one or a group of adornments or text items. Multilayer symbols are accessed through multilayer symbol classes and follow the ArcGIS Pro symbol model. These symbols can come from a mobile style file, a dictionary renderer, a mobile map package, and a feature service that uses what is called advanced symbology in other parts of the ArcGIS platform. You can also build your own multilayer symbols for points, lines, and polygons in the map. This is also known as advanced symbols and multilayer (advanced) symbols. This differs from simple symbol.
network	A set of features that are typically interconnected to each other to represent streets for a region (in a routing network) or assets (such as pipes and wires in a utility network or poles and cabinets in a structure network) for a distribution system or transmission system for a public utility, such as a gas system, water system, or electric grid. Also see subnetwork.
network source (`UtilityNetworkSource`)	A feature table whose features comprise one of a utility network's datasets. The features are used to build the network index. A utility network has multiple network sources containing related information. Some network sources contain devices appropriate for each domain network. For example, one network source for an electric utility network is a feature table named "Electric Distribution Device" representing electric devices like switches and transformers. All utility networks have network sources representing physical structures such as "Electric Support Pole" or "Electric Substation Boundary". Each network source has asset groups and types. Also known as utility network source.
nonspatial table	In ArcGIS Runtime, a table with no geometry information. It can store descriptive information, but because it doesn't store a geographical component, its features cannot be drawn on a map. In some areas of the ArcGIS Platform, a nonspatial table is known as a table. You may also know this as a nonfeature table. For more information, see Layers and tables.
offline	In ArcGIS Runtime, the state of having no network connection to ArcGIS Online or ArcGIS Enterprise.
on-demand workflow	An offline workflow that allows an ArcGIS Runtime app to generate and download a mobile map package from an online map. The mobile map package content is determined by the ArcGIS Runtime app using parameters such as the area of interest, whether to include related tables, the max and min scale. See Take a map offline - on-demand for more information.
on-the-fly reprojection	Refers to taking unprojected datasets (in other words, data that is stored in latitude and longitude) and transforming its visual representation to make it appear to match other datasets that are projected. For example, as data is being added to a map with a different spatial reference, a temporary data transformation to the map's spatial reference takes place.
operational layer	A map layer with which users can interact. Typically, an operational layer is vector based and is editable by users. However, it can also be tiled data that can be queried against. See Layers for more information.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
optimized route	The least cost path that will visit all specified stops. The order specified by the user is not necessarily followed. This is also known as the traveling salesman problem or traveling salesperson problem/solution. This is also known as optimized directions. For comparison, see simple route.
origin feature	A feature in an origin table that is associated with a destination feature in a destination table. You create associations such as this for a variety of reasons, such as to view information in one table for features in another. For more information, see Relate features in this ArcGIS Runtime guide or Essentials of relating tables.
origin table	A table that has been associated with another table, a destination table, using a key. You associate two tables for a variety of reasons, such as to view information in one table for features in another. Origin table is also known as a source table. For more information, see Relate features in this ArcGIS Runtime guide or Essentials of relating tables.
package	A set of items, such as a map and its referenced data, that ArcGIS Desktop bundles into a single file on your local machine so that the items can be easily transferred from user to user or provisioned onto a device. This is especially useful for disconnected apps.
planar measurement	Unlike geodetic measurement, which use the curvature of the earth's surface, planar measurements are made on a flat surface. When using geographic coordinates, planar measurements must cut through the curved surface of the earth, causing the measurement to be shorter than the true distance. This error becomes greater as the length of the line increases. If using a projected coordinate system, the properties of the underlying projection greatly influence the accuracy of measurements in a given area of the map.
point cloud	A usually large collection of points in 3D space representing locations of points on real-world surfaces and objects, collected with a 3D scanning device such as lidar. Locations of points on the ground, buildings, forest canopy, highway overpasses, and anything else encountered during the 3D scan survey make up the point cloud.
point cloud scene layer	A scene layer that provides high-performing display of large volumes of symbolized and filtered point cloud data.
point scene layer	A type of layer that provides high-performing display point features in a scene view (3D view) based on scale, distance, and threshold parameters associated with a viewpoint. Point scene layers are generated from point feature layers.
pop-up	A visual element on the map used to view and edit attributes and attachments associated with features or graphics in a layer on a map. While similar to attribute information that displays in an identify task, pop-ups differ because they allow you to use aliases for the field names and to have greater control over the display. A pop-up can be presented modally, in full screen, or within any other view. A pop-up can display raster cell information from a raster layer. A pop-up can display inside a callout using a callout's custom view.
portal	A term used to generically refer to ArcGIS Online, ArcGIS Enterprise, or both. These are websites that provide a framework to manage, share, and secure geographic assets, such as data, maps, apps, and services.
portal item	A web map, layer (feature, map, and image service layers), app (web and mobile apps whose content is provided by web maps), tool, or data file that you add to a portal.
preplanned workflow	An offline workflow that allows an ArcGIS Runtime app to download a mobile map package for a map area as specified by the author of an online map. See Take a map offline - preplanned for more information.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
projection	A projected coordinate system based on a map projection such as transverse Mercator, Albers equal area, or Robinson, all of which (along with numerous other map projection models) provide various mechanisms to project maps of the earth's spherical surface onto a two-dimensional Cartesian coordinate plane. Projected coordinate systems are sometimes referred to as map projections. Also see coordinate system.
provision	To provide data and other resources to an app by installing those resources with the app.
raster	A matrix of cells (or pixels) organized into rows and columns (or a grid) where each cell contains a value representing information, such as temperature. Rasters include digital aerial photographs, imagery from satellites, digital pictures, and scanned maps. For general information on rasters, see What is raster data? in the ArcGIS help.
raster layer	A layer type that allows you to display raster data in your app. A raster layer can render raster data from any type of raster. You can add it to a map as either a basemap or an operational layer.
real-scale AR	A type of augmented reality (AR) in which scene content is overlaid onto its real-world physical position. Context aids, such as a basemap, are hidden; the camera feed provides the context. This is also known as world-scale AR, full-scale AR, real-world AR, and planet-scale AR.
reference layer	A layer of information in a map that provides context to a location, such as labels for place-names, transportation routes, or other features of reference. Typically a reference layer is part of a basemap and displays at the top of the map. For more information on layer order, see Create a basemap in the Build a new map topic.
related feature	A feature in one table that has been associated with a feature in a different table using a key. See Relate features for more information.
related table	A table that has a relationship to another table. The relationship, created using ArcGIS Desktop, is made possible by a key that is common to both tables. A related table can be either a spatial table or a nonspatial table.
renderer	An object that determines how features in a layer or graphics in an overlay should be drawn (rendered) on the display. The renderer then draws them using symbols. See Symbols and renderers for more information.
rule engine	A tool that assembles military symbols using data in the style file and rules from the military specification you provide.
Runtime components	In ArcGIS Runtime SDK for Java, Qt, and .NET (Desktop), a collection of files containing various parts of ArcGIS Runtime functionality, installed and used on your development machine and deployed with the apps that you create. The full collection, which is installed to your development machine when you install an ArcGIS Runtime SDK, includes the required Runtime core components and optional components such as those for Local Server and advanced symbology. When you prepare an app for deployment, the Deployment Builder tool helps ensure you deploy only the parts of the collection that are required by the app you build.
Runtime core	Client files (`.dll` or `.so`) that must be deployed with all ArcGIS Runtime apps. For Java, Qt, and .NET versions, you can use the Deployment Builder wizard to choose which Local Server capabilities you want to deploy so they can be bundled with the Runtime core files for you.
scene view	A class (`SceneView`) that represents the 3D view tier in an MVC architecture. While a scene specifies how the three-dimensional geographic data is organized, a scene view renders the data three-dimensionally and allows users to interact with it.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
scene	A 3D representation of spatial relationships of entities within an area. In ArcGIS Runtime, a scene works together with a scene view to provide 3D visualization of geographic data on a screen. A scene specifies how the geographic data is organized; a scene view renders the data and allows users to interact with it.
Screen overlay (KML)	An image in a KML layer that is anchored to the screen (as opposed to the map). Screen overlays are generally used to display things such as titles, company logos, legends, and so on.
SDK	Software development kit. A collection of documentation, sample code, and sample apps to help a developer use an API to build apps.
service area	A region that encompasses all streets that are accessible from a given point within a given time period.
service feature table	A feature table created from a URL to an ArcGIS Feature service's layer or table. The service feature table has different request modes that affect how data is populated, queried, and cached on the client.
shapefile	A vector data storage format for storing the location, shape, and attributes of geographic features. A shapefile is stored in a set of related files.
side-by-side development	A capability that allows you to develop with different versions of the same SDK on the same machine.
sideload (sideloading)	The process of transferring data between two local devices, in particular between a computer and a mobile device such as a mobile phone or tablet.
simple relationship	A relationship between two tables in which features in the destination table are independent of features in the origin table. For example, a transformer and an electric pole may be related but they can also exist on their own. Deleting an origin feature resets the `keyField` of the relationship to NULL in the destination feature.
simple route	The least cost path that will visit all specified stops in the order specified by the user. This is also known as simple directions. For comparison, see optimized route.
simple symbol	Simple symbols follow the web map specification. You can create and work with them through the simple symbol classes in the API. These are also the symbols you get from web maps or from feature services when advanced symbology is turned off. Simple symbols can be created for points (marker symbols), lines (line symbols), and polygons (fill symbols). Each of the simple symbol types provides an enumeration of predefined styles that can be applied to the symbol. For comparison, see multilayer symbol.
sketch editor	Allows users to interactively sketch geometries on the view. It can sketch point, polygon, and polyline geometries from scratch, modify existing geometries, insert and remove vertices, undo and redo changes, and so on.
spatial reference	A coordinate-based local, regional, or global system used to precisely locate geographical entities. It defines the coordinate system used to relate map coordinates to locations in the real world. Spatial references ensure that spatial data from different layers or sources can be integrated for accurate viewing or analysis. To define a spatial reference, use either a well-known ID (WKID), which is also known as a spatial reference ID or SRID, or a full text definition (referred to as well-known text, WKT). Also see Coordinate systems and transformation.
spatial table	A table with geometry information; its features can be displayed on a map.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
static rendering mode	One of two modes for rendering a graphics overlay. Use this mode for static graphics, complex geometries, and very large numbers of polygons. Volume of graphics has little impact on frame render time (scales well) and pushes a constant GPU payload. However, rendering graphic updates is CPU/system memory intensive, which can have an impact on device battery life. The other rendering mode is called dynamic. For a comparison, see the definition for dynamic rendering mode.
structure network	A network that represents the features that do not carry a resource (such as cabinets and poles) in a utility network. All domain networks in a utility network share a common structure network.
style	A style is a server or file-defined option for how content should be rendered in a map. WMS defines styles as a concept that applies to layers. KML defines styles as a concept that applies to placemarks.
style file	A `.stylx` file you can create in ArcGIS Pro that contains symbol primitives such as the symbol frame, the lines that make up inner icons, and so on.
sublayer	One of several layers that are part of a group layer. For example, a map service layer is a type of group layer that can contain one or more sub image layers.
subnetwork	A topological subpart within a tier where all the connected features are defined by the same subnetwork controller or controllers. Examples of subnetworks are circuits in electric domain networks and pressure zones in gas and water domain networks. To create a subnetwork, a subnetwork controller is set, the network topology is validated, and the subnetwork is updated. Good network management depends on the reliability of the paths in a network (or subnetworks). The management of these subnetworks allows organizations to optimize the delivery of resources and track the status of a network.
subnetwork controller	A type of utility network feature from which a resource is delivered or collected, such as a circuit breaker (electric), town border stations (gas), or water towers (water). A subnetwork controller type is defined for each domain network, and when a domain network's tier is configured, the tier is defined to use one or more of the controllers in the domain network. Certain asset group and asset type features in the device feature class are configured to allow features to be set as subnetwork controllers; this done by assigning a network category. Subnetwork controllers are set for device features using a specific terminal and are used as start or end points for trace analysis.
subnetwork trace	A trace that discovers all features participating in a subnetwork. This trace type is useful for validating whether subnetworks, such as circuits or zones, are defined or edited appropriately. The trace begins at one or more starting points and spans outward along connected features to find subnetwork controllers that are traversable. A subnetwork trace stops when it encounters a barrier, when it encounters a feature that is not traversable, or when there are no more connected features. For details, see this guide's topic on performing traces or see ArcGIS Pro's help topic on subnetwork trace.
subsurface	The area below the surface of the earth. In a scene (in 3D), you can map content located either above or below the earth's surface (or both), such as earthquakes, mineral deposits, wells, pipes, or geologic strata. You can rotate the view to observe it from different viewpoints/angles within the scene. You can set your viewpoint to be below the earth's surface and you can adjust the transparency of the surface.
subsurface navigation	Zooming, panning, and tilting the scene from a viewpoint located below the earth's surface. Also see subsurface.
subtype	An asset group in a utility network.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
subtype feature layer	A feature layer that allows symbology and other layer properties to set on a per-subtype basis.
subtype group layer	See subtype feature layer.
symbol	A symbol defines all the nongeographic aspects of a graphic or feature's appearance, including color, size, border, and transparency. You can use symbols to create a renderer for graphics overlays or feature layers. You can also apply a symbol directly to individual graphics when you create them. For more information on using symbols in ArcGIS Runtime, see .Symbols and renderers.
symbol dictionary	A set of symbol primitives and a rule engine that together allow you to use military symbols in your app. The military symbols adhere to military symbol specifications such as MIL-STD-2525C. The symbol primitives are inside a style file. To display military symbols, you associate the layer/overlay's dictionary renderer to a symbol dictionary.
system	A term with many meanings in computer technology, from a personal computer to a platform or framework of computers on. For public utility ArcGIS customers, a system refers to a collection of assets that make up a public utility, such as a city's water distribution system or a gas company's gas distribution system. In ArcGIS these utility systems are known asa utility networks.
table	In ArcGIS Runtime, a data source for ArcGIS data that may or may not contain geometry information. In some areas of the ArcGIS platform, table refers to a table with no geometry information. For more information on using tables in ArcGIS Runtime, see Layers and tables.
task	A class bound to online or offline data or services that provides methods to perform asynchronous operations using those data or services. For details, see Tasks and jobs.
terminal	In ArcGIS, a class that represent physical devices in the field. Each terminal can have ports that model flow in and out. Each device can contain many terminals, each of which can be interconnected. Terminal configurations model these interconnections by specifying the flow path through terminals. For example, consider a bypass switch with an attached voltage regulator. The bypass switch exists so the voltage regulator may be bypassed (for inspection and service, for example) without disrupting the electrical flow that normally passes through the voltage regulator. In normal operation, the blades inside the bypass switch connect certain device terminals together, while in bypass operation, other terminals are connected instead, so that the electricity flows along the desired path within the bypass switch. For more information see the device terminals discussion in ArcGIS Pro documentation.
texture	An image file applied to a surface to assist in visual discrimination for the viewer and impart a sense of visual depth (3D).
texture compression format	The compression format used to compress an image used as a texture. A typical compression technique and file format for textures is JPG. However, for mobile platforms, the ETC2 compression technique consumes less memory and improves download time and overall performance. For more information, see Author a mobile scene package for ArcGIS Runtime SDK.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
tier	1. A level in the MVC programming pattern. 2. In ArcGIS, a level in the hierarchy of a utility network that helps model the hierarchy of how a utility network delivers a resource such as gas, electricity, or water. For example, an electric distribution system can be subdivided into subtransmission, medium voltage, and low voltage tiers. Each utility network can contain up to 63 tiers defined in a one or more domain networks. In ArcGIS Runtime it's represented by the `UtilityTier` class. A tier can also define the topological organization (radial or mesh) of subnetworks that all share the same properties and adhere to the same restrictions. You can decide whether to trace within these tiers or across tiers using the tier's trace configuration. This configuration can support traversability, propagators, and function barriers For more information see the tiers topic in ArcGIS Pro documentation or the trace configuration discussion in this guide.
tile	An image, often a graphics file (for example, a `.jpg` file), typically stored in a directory known as a cache. The image is part of a set of tiles that, conceptually, are pieces of a bigger map. How the tiles fit into the bigger map, along with other information, is defined in a tiling scheme. Two tile categories are as follows: • Tiles in dynamic layers are created on the fly and can be stored in a cache on the client machine or on the server machine. Where the user pans and zooms affects which tiles are created in this scenario, so technically, the tiles may not form a bigger, complete map when put together. • Tiles in tile cache layers are created before users view the map, often by a developer or GIS data administrator. These tiles are known as preprocessed tiles. Also see tile cache.
tile cache	A directory that contains tiles of a map extent at specific levels. The directory can be local to a desktop app or to a client app in a client/server configuration. Also see tile.
tiled layer	A layer displayed by assembling tiles (rectangular sections) into a continuous layer. The tiles are either raster image tiles or vector tiles; they're generated into a tile cache before they're available for use. (Compare this to a dynamic layer, which renders itself on the fly.) Tiled layers are often used for basemaps. You may see the term tiled layer used to refer specifically to the raster tiled layer type because vector tiled layers are newer than raster tile layers. Also see vector tiled layer.
tile package	A tiled layer that's been bundled into a single `.tpk/.tpkx` file. The file contains a tile cache of the data and metadata about the layer, packaged into a single, portable file. You can add a tile package to an ArcGIS Runtime app using the local tiled layer class, allowing you to share tile layers via regular file sharing methods (email, FTP, and so on) and through ArcGIS Online. Tile packages are ideal in disconnected environments where access to local data is required, and are ideal for displaying basemaps.
trace	An action that analyzes the paths in a utility network and returns selected features based on connectivity or traversability from the specified starting points. For example, a trace can show an electric utility employee every asset that is connected to a source or find all features upstream or downstream from a selected point.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
transformation	The task of moving your data between different geographic coordinate systems. You may, for example, have some data in WGS84 that was collected from a GPS reading. However, your map may be in a different spatial reference, such as British National Grid, which is based on a different geographic coordinate system, OSGB 1936. To convert the data from WGS84 to British National Grid, you need to apply a transformation as well as a projection. Many transformations are available, depending on the geographic area your data comes from. In this SDK, transformations are performed using equation-based transformation methods or grid-based transformation methods. For details about transformations, see Spatial references.
unique value renderer	A renderer that allows you to use one or more values in a field to specify how features with that same value (or values) should be rendered.
upstream trace	In a source-based network (gas or electric), an upstream trace is against the flow and toward the source, such as a circuit breaker or generator (subnetwork controller). In a sink-based network (sewer or storm water), an upstream trace is against the flow and away from the sink such as a sewage treatment (subnetwork controller). For details, see this guide's topic on performing traces or see ArcGIS Pro's help topic on upstream traces.
utility element (`UtilityElement`)	An entity in a utility network that can correspond to a feature or a piece of a feature (for example, a terminal inside a device feature or an edge within a linear feature). Utility elements are used for a range of purposes, including the following: • Specifying starting points and barriers for tracing • Defining the "from" and "to" sides of associations • Returning trace results A utility element includes a reference to a feature inside a utility network source plus a terminal (if applicable).
utility network (`UtilityNetwork`)	In ArcGIS Runtime, a feature service that represents a utility system such as a water, gas, or electric system.A utility network provides capabilities to visualize, edit, and analyze utility assets and data. It includes one or more domain networks (gas, water, electric, or other) plus a structure network. An organization specifies the set of domain networks that it manages when it configures a utility network. It is possible to define associations across domain networks and enable tracing analysis across those domains. For example, you can perform electric tracing analysis from transmission to distribution levels with a utility network using electric transmission and electric distribution domain networks. For more information, see the utility networks topic.
utility network definition (`UtilityNetworkDefinition`)	In ArcGIS Runtime, metadata in the `UtilityDomainNetwork` class used for a utility network's feature service. This class serves as the entry point to the metadata for all elements in the utility network, such as domain networks, network sources, and terminal configurations.
utility network source	See network source.
vector tiled layer	A layer that's similar to tiled layers (made with raster tiles) but requires less space and differs in the way cartography is delivered. Instead of pixels, cartography is delivered with 2D points that define lines, polygons, and the locations of labels and marker symbols. Cartography is rendered at runtime, so differences between levels of detail appear more continuous than with raster tiles. The file format is binary and conforms to the Mapbox vector tile specification.

Copyright © 1995-2020 Esri. All rights reserved.

Term	Definition
vector tile package	A vector tile layer that's been bundled into a single file (a `.vtpk` file). The file contains all the tile data files, the service definition, a style sheet, the fonts, and the symbol markers required to display the map. It can be downloaded from an ArcGIS Online vector tile service.
vertex	A point that stands alone or makes up part of a geometry. Vertices that make up a geometry can be connected, one to the next, in a linear order.
viewpoint	A viewpoint is the visible area and view location of a GeoView. It is what the user sees when viewing the map. It can be used to define and control the position, extent, scale, and rotation of the view.
viewshed	The locations visible from one or more specified points or lines. Viewshed maps are useful for such applications as finding well-exposed places for communication towers, or hidden places for parking lots. You can create viewsheds using the Standard license level of ArcGIS Runtime SDKs.
wander extent	The factor of map extent within which the location symbol may move before triggering auto-panning to recenter the map on the current location.
web feature service (WFS)	WFS is an Open Geospatial Consortium (OGC) standard for feature services. WFS provides feature data on demand and provides methods for querying data using spatial and nonspatial queries.
web map	An interactive display of geographic information. A web map can be described as a collection of geographic layers, behaviors, and tools. It can contain a basemap, a set of data layers (many of which include interactive pop-up windows with information about the data), and an extent. For more detailed information, see Web maps.
web map service (WMS)	WMS is an Open Geospatial Consortium (OGC) standard that defines image-based map services. WMS services provide map images for specific areas within a map on demand. Images include prerendered symbology and may be rendered in one of several named styles if defined by the service.
web scene	A web scene allows you to visualize and analyze geographic information in an intuitive and interactive 3D environment. Each web scene typically contains a basemap, an elevation source, a collection of geographic layers, and a camera view point. Web scenes can be created, published, and consumed in ArcGIS Pro and ArcGIS Online. For more detailed information, see Web scenes.

Copyright © 1995-2020 Esri. All rights reserved.

Coordinate systems and transformations

These PDF documents contain tables of well-known IDs (WKIDs) and other details about coordinate systems, datum transformations, and units. For details on how to use this information with ArcGIS Runtime, see Spatial references.

- Geographic coordinate systems (PDF format)
- Projected coordinate systems (PDF format)
- Geographic transformations (PDF format)

Related topics

Spatial references

Copyright © 1995-2020 Esri. All rights reserved.

Local Server geoprocessing tools support

Local services allow you to deploy services that run locally with your app to provide functionality like geocoding, geoprocessing, routing, and so on. In ArcGIS Runtime SDK for .NET, local services are only available for Windows Presentation Framework (WPF) apps. An app built for another ArcGIS Runtime for .NET platform can use such functionality from an online service, but not locally.

This topic does not apply for the iOS platform.

Copyright © 1995-2020 Esri. All rights reserved.

Platform error codes

Internal API errors are handled a few different ways depending on the type of error and the general exception handling patterns common to the platform. Some APIs use exceptions to manage errors. Error codes are documented here for reference, but for clarity refer to the specific API references for methods that throw exceptions or return error information. In most cases, the API will defer error functionality to the objects and services available on the native platform. For example,

- Android/Java: in general, the API defers error functionality to `ArcGISRuntimeException`.
- iOS/macOS: in general, the API defers functionality to `NSError`.
- .NET: in general, the API defers error functionality to `ArcGISRuntimeException`.
- Qt: in general, the API manages error information using an `Error` object.

There are distinct classes of errors, depending on the service invoked. As the architecture dictates, errors may originate from the API, the runtime core supporting the API, or the service (the servers the API connects to implementing certain services.) Check the error domain if this is information useful to your app. In most cases you only need the error code to determine the cause of the error and possible remedy.

Error domains

The API exposes several error domains. A domain is a segregated area of the API, usually bound to a framework, internal library, or module of functionality. The error domain helps focus the error to a specific area of functionality within the API.

- ArcGISRuntime
- ArcGISServer
- ArcGISPopup

Handling errors

There are different coding patterns used to handle errors that are specific to each API. Refer to the specific API documentation to understand how that API will communicate an error response. There are three patterns of error response:

Return codes: sometimes an API returns an error value or `null`. You test the return value and handle it according to your application implementation requirements.

Callback, lambda, or delegate function: certain APIs allow the caller to set an asynchronous callback function that the internal API calls upon certain defined events occurring. It is possible in these callbacks to receive an error. The API defines hows the error is represented in the callback. It is up to your application to handle the error.

Exceptions: some APIs generate an exception you should handle. There are many ways of receiving an exception: try/catch, error block/closure, error callback, or signal.

Error codes

Error codes are defined here as integers. In order to safeguard your code from comparing against these numbers directly we suggest some form of abstraction, such as an errors enumeration or error class. We will do our best to never change a defined error code once the SDK is officially released, but we reserve the right to make changes. Please consult the release notes regarding any changes to these values.

Copyright © 1995-2020 Esri. All rights reserved.

ArcGISPopup errors

API errors are client-side exceptions that usually occur due to incorrect method parameters, object state dependencies, or network conditions.

Error Code	Description
11000	Invalid attributes or geometry on the associated GeoElement.
11001	Null value is not allowed.
11002	Value is out of range.
11003	Value exceeds maximum field length.

ArcGISRuntime errors

Internal runtime errors are thrown by the core library when things go wrong client-side. Where possible the details of the exception are described. In many instances, the details of the exception are dependent on the context that generated the exception. In these cases, refer to the specific API that generated the error for more information.

Error Code	Description
-1	This is the catch all for unknown errors.
0	This is a success and not an error.
1	Null pointer exception.
2	Invalid argument exception.
3	Not implemented exception.
4	Out of range exception.
5	Invalid access exception.
6	Illegal state exception.
7	Not found exception.
8	Exists exception.
9	Timeout exception.
10	Regular expression exception.
11	Property not supported exception.
12	No permission exception.
13	File exception.
14	File not found exception.
15	Invalid call exception.
16	IO exception.
17	User canceled exception.
18	Internal error exception.
19	Conversion failed exception.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
20	No data.
21	Attempted to use invalid JSON.
22	Error was propagated.
23	Attempted to use invalid XML.
24	Object is already owned.
25	Reserved for use by Qt.
26	The resource is past its expiry date.
27	An unexpected null was returned from a property or method which is expected to be non-nullable.

SQL errors

Internal errors related to the SQL storage engine.

Error Code	Description
1001	SQLite error exception.
1002	SQLite Internal exception.
1003	SQLite Perm exception.
1004	SQLite Abort exception.
1005	SQLite Busy exception.
1006	SQLite Locked exception.
1007	SQLite NoMem exception.
1008	SQLite Read only exception.
1009	SQLite Interrupt exception.
1010	SQLite IO Error exception.
1011	SQLite Corrupt exception.
1012	SQLite Not found exception.
1013	SQLite Full exception.
1014	SQLite Can't open exception.
1015	SQLite Protocol exception.
1016	SQLite Empty exception.
1017	SQLite Schema exception.
1018	SQLite Too big exception.
1019	SQLite Constraint exception.
1020	SQLite Mismatch exception.
1021	SQLite Misuse exception.
1022	SQLite Nolfs exception.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
1023	SQLite Auth exception.
1024	SQLite Format exception.
1025	SQLite Range exception.
1026	SQLite Notadb exception.
1027	SQLite Notice exception.
1028	SQLite Warning exception.
1029	SQLite Row exception.
1030	SQLite Done exception.

Geometry errors

Internal errors related to the geometry engine.

Error Code	Description
2000	Geometry Exception.
2001	Geometry Corrupted geometry exception.
2002	Geometry Empty geometry exception.
2003	Geometry Math singularity exception.
2004	Geometry Buffer is too small exception.
2005	Geometry Invalid shape type exception.
2006	Geometry Projection out of supported range exception.
2007	Geometry Non simple geometry exception.
2008	Geometry Cannot calculate geodesic exception.
2009	Geometry Notation conversion exception.
2010	Geometry Missing grid file exception.

Geodatabase errors

Internal errors related to geodatabase queries.

Error Code	Description
3001	Geodatabase Value out of range exception.
3002	Geodatabase Data type mismatch exception.
3003	Geodatabase Bad XML exception.
3004	Geodatabase Database already exists exception.
3005	Geodatabase Database does not exist exception.
3006	Geodatabase Name longer than 128 characters exception.
3007	Geodatabase Invalid shape type exception.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
3008	Geodatabase Raster not supported exception.
3009	Geodatabase Relationship class one to one exception.
3010	Geodatabase Item not found exception.
3011	Geodatabase Duplicate code exception.
3012	Geodatabase Missing code exception.
3013	Geodatabase Wrong item type exception.
3014	Geodatabase Id field not nullable exception.
3015	Geodatabase Default value not supported exception.
3016	Geodatabase Table not editable exception.
3017	Geodatabase Field not found exception.
3018	Geodatabase Field exists exception.
3019	Geodatabase Cannot alter field type exception.
3020	Geodatabase Cannot alter field width exception.
3021	Geodatabase Cannot alter field to nullable exception.
3022	Geodatabase Cannot alter field to editable exception.
3023	Geodatabase Cannot alter field to deletable exception.
3024	Geodatabase Cannot alter geometry properties exception.
3025	Geodatabase Unnamed table exception.
3026	Geodatabase Invalid type for domain exception.
3027	Geodatabase Min max reversed exception.
3028	Geodatabase Field not supported on relationship class exception.
3029	Geodatabase Relationship class key exception.
3030	Geodatabase Value is null exception.
3031	Geodatabase Multiple SQL statements exception.
3032	Geodatabase No SQL statements exception.
3033	Geodatabase Geometry field missing exception.
3034	Geodatabase Transaction started exception.
3035	Geodatabase Transaction not started exception.
3036	Geodatabase Shape requires z exception.
3037	Geodatabase shape requires m exception.
3038	Geodatabase Shape no z exception.
3039	Geodatabase Shape no m exception.
3040	Geodatabase Shape wrong type exception.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
3041	Geodatabase Cannot update field type exception.
3042	Geodatabase No rows affected exception.
3043	Geodatabase Subtype invalid exception.
3044	Geodatabase Subtype must be integer exception.
3045	Geodatabase Subtypes not enabled exception.
3046	Geodatabase Subtype exists exception.
3047	Geodatabase Duplicate field not allowed exception.
3048	Geodatabase Cannot delete field exception.
3049	Geodatabase Index exists exception.
3050	Geodatabase Index not found exception.
3051	Geodatabase Cursor not on row exception.
3052	Geodatabase Internal error exception.
3053	Geodatabase Cannot write geodatabase managed fields exception.
3054	Geodatabase Item already exists exception.
3055	Geodatabase Invalid spatial index name exception.
3056	Geodatabase Requires spatial index exception.
3057	Geodatabase Reserved name exception.
3058	Geodatabase Cannot update schema if change tracking exception.
3059	Geodatabase Invalid date exception.
3060	Geodatabase Database does not have changes exception.
3061	Geodatabase Replica does not exists exception.
3062	Geodatabase Storage type not supported exception.
3063	Geodatabase Replica model error exception.
3064	Geodatabase Replica client gen error exception.
3065	Geodatabase Replica no upload to acknowledge exception.
3066	Geodatabase Last write time in the future exception.
3067	Geodatabase Invalid argument exception.
3068	Geodatabase Transportation network corrupt exception.
3069	Geodatabase Transportation network file IO exception.
3070	Geodatabase Feature has pending edits exception.
3071	Geodatabase Change tracking not enabled exception.
3072	Geodatabase Transportation network file open exception.
3073	Geodatabase Transportation network unsupported exception.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
3074	Geodatabase Cannot sync copy exception.
3075	Geodatabase Access control denied exception.
3076	Geodatabase Geometry outside replica extent exception.
3077	Geodatabase Upload already in progress exception.
3078	Geodatabase Database is closed exception.
3079	Domain Already Exists exception.
3080	Geometry type not supported exception.

Geocode errors

Internal errors related to geocoding, address, and reverse address lookup.

Error Code	Description
4001	Geocode Unsupported file format exception.
4002	Geocode Unsupported spatial reference exception.
4003	Geocode Unsupported projection transformation exception.
4004	Geocode Geocoder creation exception.
4005	Geocode Intersections not supported exception.
4006	Geocode Uninitialized geocode task exception.
4007	Geocode Invalid locator properties exception.
4008	Geocode Required field missing exception.
4009	Geocode Cannot read address exception.
4010	Geocode Geocoding not supported exception.

Network Analyst errors

Internal errors related to Network Analyst and routing tasks.

Error Code	Description
5001	Network Analyst Invalid route settings exception.
5002	Network Analyst No solution exception.
5003	Network Analyst Task canceled exception.
5004	Network Analyst Invalid network exception.
5005	Network Analyst Directions error exception.
5006	Network Analyst Insufficient number of stops exception.
5007	Network Analyst Stop unlocated exception.
5008	Network Analyst Stop located on non traversable element exception.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
5009	Network Analyst Point barrier invalid added cost attribute name exception.
5010	Network Analyst Line barrier invalid scaled cost attribute name exception.
5011	Network Analyst Polygon barrier invalid scaled cost attribute name.
5012	Network Analyst Polygon barrier invalid scaled cost attribute value.
5013	Network Analyst Polyline barrier invalid scaled cost attribute value.
5014	Network Analyst Invalid impedance attribute exception.
5015	Network Analyst Invalid restriction attribute exception.
5016	Network Analyst Invalid accumulate attribute exception.
5017	Network Analyst Invalid directions time attribute exception.
5018	Network Analyst Invalid directions distance attribute exception.
5019	Network Analyst Invalid attribute parameters attribute name exception.
5020	Network Analyst Invalid attributes parameters parameter name exception.
5021	Network Analyst Invalid attributes parameters value type exception.
5022	Network Analyst Invalid attribute parameters restriction usage value exception.
5023	Network Analyst Network has no hierarchy attribute exception.
5024	Network Analyst No path found between stops exception.
5025	Network Analyst Undefined input spatial reference exception.
5026	Network Analyst Undefined output spatial reference exception.
5027	Network Analyst Invalid directions style exception.
5028	Deprecated. Network Analyst Invalid directions language exception.
5029	Network Analyst Directions time and impedance attribute mismatch exception.
5030	Network Analyst Invalid directions road class attribute exception.
5031	Network Analyst Stop can not be reached.
5032	Network Analyst Stop time window starts before unix epoch exception.
5033	Network Analyst Stop time window is inverted exception.
5034	Walking mode route too large exception.
5035	Stop has null geometry exception.
5036	Point barrier has null geometry exception.
5037	Polyline barrier has null geometry exception.
5038	Polygon barrier has null geometry.
5039	Online route task does not support search_where_clause condition exception.
5040	Network Analyst Insufficient number of facilities exception.
5041	Network Analyst Facility has null geometry exception.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
5042	Network Analyst Facility has invalid added cost attribute name exception.
5043	Network Analyst Facility has negative added cost attribute exception.
5044	Network Analyst Facility has invalid impedance cutoff exception.
5045	Network Analyst insufficient number of incidents exception.
5046	Network Analyst Incident has null geometry exception.
5047	Network Analyst Incident has invalid added cost attribute name exception.
5048	Network Analyst Incident has negative added cost attribute exception.
5049	Network Analyst Invalid target facility count exception.
5050	Network Analyst Incident has invalid impedance cutoff exception.
5051	Network Analyst start time is before Unix epoch exception.
5052	Network Analyst Invalid default impedance cutoff exception.
5053	Network Analyst Invalid default target facility count exception.
5054	Network Analyst Invalid polygon buffer distance exception.
5055	Network Analyst Polylines cannot be returned.
5056	Network Analyst Solving non time impedance, but time windows applied.
5057	One or more stops have unsupported type.
5058	Network Analyst Route starts or ends on a waypoint.
5059	Network Analyst Reordering stops (Traveling Salesman Problem) is not supported when the collection of stops contains waypoints or rest breaks.
5060	Network Analyst The waypoint contains time windows.
5061	Network Analyst The waypoint contains added cost attribute.
5062	Network Analyst The stop has unknown curb approach.
5063	Network Analyst The point barrier has unknown curb approach.
5064	Network Analyst The facility has unknown curb approach.
5065	Network Analyst The incident has unknown curb approach.
5066	Network dataset has no directions attributes.
5067	Desired direction language not supported by platform.
5068	Route result requires re-solving with current route task.
5069	Input route result does not have directions.
5070	Input route result does not have stops.
5071	Input route result doesn't have the route with passed route index.
5072	Input remaining destinations count doesn't match with input routes stops collection.

Copyright © 1995-2020 Esri. All rights reserved.

JSON parsing errors

Errors that may occur when dealing with JSON.

Error Code	Description
6001	JSON parser invalid token exception.
6002	JSON parser invalid character exception.
6003	JSON parser invalid unicode exception.
6004	JSON parser invalid start of JSON exception.
6005	JSON parser invalid end of pair exception.
6006	JSON parser invalid end of element exception.
6007	JSON parser invalid escape sequence exception.
6008	JSON parser invalid end of field name exception.
6009	JSON parser invalid start of field name exception.
6010	JSON parser invalid start of comment exception.
6011	JSON parser invalid decimal digit exception.
6012	JSON parser invalid hex digit.
6013	JSON parser expecting null exception.
6014	JSON parser expecting true exception.
6015	JSON parser expecting false exception.
6016	JSON parser expecting closing quote exception.
6017	JSON parser expecting not a number exception.
6018	JSON parser expecting end of comment exception.
6019	JSON parser unexpected end of data exception.
6020	JSON object expecting start object exception.
6021	JSON object expecting start array exception.
6022	JSON object expecting value object exception.
6023	JSON object expecting value array exception.
6024	JSON object expecting value int32 exception.
6025	JSON object expecting integer type exception.
6026	JSON object expecting number type exception.
6027	JSON object expecting value string exception.
6028	JSON object expecting value bool exception.
6029	JSON object iterator not started exception.
6030	JSON object iterator is finished exception.
6031	JSON object key not found exception.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
6032	JSON object index out of range exception.
6033	JSON string writer JSON is complete exception.
6034	JSON string writer invalid JSON input exception.
6035	JSON string writer expecting container exception.
6036	JSON string writer expecting key or end object exception.
6037	JSON string writer expecting value or end array exception.
6038	JSON string writer expecting value exception.

Internal errors

Errors related to the internal ArcGISRuntime domain.

Error Code	Description
7001	The spatial reference is missing.
7002	The initial viewpoint is missing.
7003	Expected a different response to the request.
7004	The Bing maps key is missing.
7005	The layer type is not supported.
7006	Cannot sync because it is not enabled.
7007	Cannot export tiles because it is not enabled.
7008	Required item property is missing.
7009	Webmap version is not supported.
7011	Spatial Reference invalid or incompatible
7012	The package needs to be unpacked before it can be used.
7013	The elevation source data format is not supported.
7014	Webscene version or viewing mode is not supported.
7015	Loadable object is not loaded when it is expected to be loaded.
7016	Scheduled updates for an offline preplanned map area are not supported.
7017	The trace operation executed by the service failed.
7018	The Arcade expression is invalid.
7019	The requested extent contains too many associations.
7020	A layer has requested more features than the service maximum feature count.

Licensing errors

Errors that may occur when working with licensing.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
8001	Unlicensed feature exception. Unlicensed feature exception.
8002	License level fixed exception. License level fixed exception.
8003	License level is already set exception. License level is already set exception.
8004	Main license is not set exception. Main license is not set exception.
8005	Unlicensed extension exception. Unlicensed extension exception.
8006	Portal user with no license exception.

Local server errors

Errors that may occur when working with local server.

Error Code	Description
9001	Local server failed to start.
9002	A local server's service failed to start.
9003	A local server's service terminated unexpectedly.
9004	The local server has failed.
9005	A local server's service has failed.

Internal library errors

Internal errors related to the use of common libraries.

Error Code	Description
10001	std::ios_base::failure exception.
10002	std::bad_array_new_length exception.
10003	std::underflow_error exception.
10004	std::system_error exception.
10005	std::range_error exception.
10006	std::overflow_error exception.
10007	std::out_of_range exception.
10008	std::length_error exception.
10009	std::invalid_argument exception.
10010	std::future_error exception.
10011	std::domain_error exception.
10012	std::runtime_error exception.
10013	std::logic_error exception.
10014	std::bad_weak_ptr exception.
10015	std::bad_typeid exception.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
10016	std::bad_function_call exception.
10017	std::bad_exception exception.
10018	std::bad_cast exception.
10019	std::bad_alloc exception.
10020	std::exception exception.

Electronic navigation charts (ENC)

Errors that may occur when working with ENC APIs.

Error Code	Description
12001	SSE 01: Self-Signed Key is invalid.
12002	SSE 02: Format of Self-Signed Key file is incorrect.
12003	SSE 03: SA-Signed Data Server Certificate is invalid.
12004	SSE 04: Format of SA Signed DS Certificate is incorrect.
12005	SSE 05: SA Digital Certificate (X509) file is not available. A valid certificate can be obtained from the IHO website or your data supplier.
12006	SSE 06: The SA-Signed Data Server Certificate is invalid. The SA may have issued a new public key or the ENC may originate from another service. A new SA public key can be obtained from the IHO website or from your data supplier.
12007	SSE 07: SA-signed DS Certificate file is not available. A valid certificate can be obtained from the IHO website or your data supplier.
12008	SSE 08: SA Digital Certificate (X509) file incorrect format. A valid certificate can be obtained from the IHO website or your data supplier.
12009	SSE 09: ENC Signature is invalid.
12010	SSE 10: Permits not available for this Data Server. Contact your data supplier to obtain the correct permits..
12011	SSE 11: Cell Permit not found. Load the permit file provided by the data supplier.
12012	SSE 12: Cell Permit format is incorrect. Contact your data supplier and obtain a new permit file.
12013	SSE 13: Cell Permit is invalid (checksum is incorrect) or the Cell Permit is for a different system. Contact your data supplier and obtain a new or valid permit file.
12014	SSE 14: Incorrect system date, check that the computer clock (if accessible) is set correctly or contact your system supplier.
12015	SSE 15: Subscription service has expired. Please contact your data supplier to renew the subscription license.
12016	SSE 16: ENC CRC value is incorrect. Contact your data supplier as ENC(s) may be corrupted or missing data.
12017	SSE 17: Userpermit is invalid (checksum is incorrect). Check that the correct hardware device (dongle) is connected or contact your system supplier to obtain a valid userpermit.
12018	SSE 18: HW_ID is incorrect format.
12019	SSE 19: Permits are not valid for this system. Contact your data supplier to obtain the correct permits.

Copyright © 1995-2020 Esri. All rights reserved.

Error Code	Description
12020	SSE 20: Subscription service will expire in less than 30 days. Please contact your data supplier to renew the subscription license.
12021	SSE 21: Decryption failed no valid cell permit found. Permits may be for another system or new permits may be required, please contact your supplier to obtain a new license.
12022	SSE 22: SA Digital Certificate (X509) has expired. A new SA public key can be obtained from the IHO website or from your data supplier.
12023	SSE 23: Non-sequential update, previous update(s) missing. Try reloading from the base media. If the problem persists, contact your data supplier.
12024	SSE 24: ENC Signature format incorrect, contact your data supplier.
12025	SSE 25: The permit for ENC<cell name> has expired. This cell may be out of date and must not be used for primary navigation.
12026	SSE 26: This ENC is not authenticated by the IHO acting as the Scheme Administrator.
12027	SSE 27: ENC<cell name> is not up to date. A new edition, reissue or update for this cell is missing and therefore must not be used for primary navigation.

Navigation

Errors that may occur when working with Navigation APIs.

Error Code	Description
13001	Service doesn't support rerouting.

Feature service edit errors

Edit errors are returned by the server after you call apply edits on a service feature table. The EditError () object in the API wraps these error codes and messages. Refer to feature service error codes for more information.

HTTP, network, and REST errors

Some APIs communicate with servers and services over the World Wide Web using HTTP, such as the low-level REST API. These services may generate standard HTTP error codes that may have meaning related to the ArcGIS platform.

Error	Error Code	Description
Bad request	400	The request sent to the server is not correct.
Unauthorized	401	Authorization to the requested resource is required.
Forbidden	403	Authorization to the requested resource is required.
Not found	404	The requested resource was not found.
Payload too large	413	The request is larger than limits defined by the server. If you're trying to upload an attachment, this error might indicate that the attachment's size exceeds the maximum size allowed.
Invalid token	498	The access token provided is invalid or expired.
Token required	499	A token was required to access the resource.

Copyright © 1995-2020 Esri. All rights reserved.

Error	Error Code	Description
Service error	500	The service was not able to fulfill the request, possibly due to invalid input, or the service may not be functioning properly.
Not implemented	501	The requested service is not implemented.

General errors

Error	Description
ArcGIS Runtime error Invalid access range Feature service is not loaded, it needs to be loaded before passed into the sync task.	Check that the offline data you're trying to access has been created correctly and make sure you understand the offline considerations described here. A mobile geodatabase (.geodatabase file) must come either from an online feature service (from ArcGIS Online or ArcGIS Server 10.x) or be side loaded from content created by ArcGIS Desktop's Create Runtime Content tool. You can't manually create an offline mobile geodatabase. For details on creating and using a mobile geodatabase 1) in a service, see the services pattern 2) in the side loading pattern, see the desktop pattern. A feature service must be sync-enabled if you want to edit it offline. Determine whether or not your service is sync-enabled by looking at the REST endpoint. A sync-enabled service has three specific REST endpoints listed at the bottom of its HTML page: 1) Create Replica 2) Synchronize Replica 3) Unregister Replica. For details on how to create a sync-enabled service on ArcGIS Online, see Managing hosted feature layers. Map services cannot be sync-enabled.

Copyright © 1995-2020 Esri. All rights reserved.

Supported languages

Language support in ArcGIS Runtime provides exception messages in the language of the computer running the app. The following languages (cultures) are supported.

- Arabic (ar)
- Bosnian (bs)
- Czech (cs)
- Danish (da)
- German (de)
- Greek (el)
- Spanish (es)
- Estonian (et)
- Finnish (fi)
- French (fr)
- Hebrew (he)
- Hindi (hi)
- Croatian (hr)
- Indonesian (id)
- Italian (it)
- Japanese (ja)
- Korean (ko)
- Lithuanian (lt)
- Latvian (lv)
- Norwegian Bokmål (nb-NO)
- Norwegian (no)
- Dutch (nl)
- Polish (pl)
- Portuguese (pt-BR, pt-PT)
- Romanian (ro)
- Russian (ru)
- Serbian (sr)
- Swedish (sv)
- Thai (th)
- Turkish (tr)
- Ukrainian (uk-UA)

Copyright © 1995-2020 Esri. All rights reserved.

- Vietnamese (vi)
- Simplified Chinese (zh-CN, zh-Hans)
- Traditional Chinese (zh-HK, zh-TW)

Copyright © 1995-2020 Esri. All rights reserved.

JSON label class properties

The available properties for defining JSON label classes are described here, along with the expected values. Most of these properties are optional, so the default values applied are also described.

allowOverlapOfFeatureBoundary

Optional string describing whether other labels are allowed to overlap this feature's edge. Applies to polygons only.

Values	Description
`allow`	Labels are allowed to overlap this polygon feature boundary.
`avoid`	Labels that would overlap will move as much possible to minimize the overlap.
`exclude`	Labels that would overlap are not placed.
Default = `allow`	

allowOverlapOfFeatureInterior

Optional string describing how much other labels are allowed to overlap this feature.

Values	Description
`allow`	Labels are allowed to overlap this feature.
`avoid`	Labels that would overlap will move as much possible to minimize the overlap.
`exclude`	Labels that would overlap are not placed.
Default = `allow`	

allowOverlapOfLabel

Optional string describing whether other labels are allowed to overlap this label.

Values	Description
`allow`	Labels are allowed to overlap this label.
`avoid`	Labels that would overlap will move as much possible to minimize the overlap.
`exclude`	Labels that would overlap are not placed.
Default = `allow`	

allowOverrun

Optional boolean to control whether a label will be visible at scales where the feature is too small for the label to fit.

Values	Description
`true`	Label can run past the ends of its line feature or edges or its polygon feature.
`false`	Label cannot run past the ends of its line feature or edges or its polygon feature.
Default = `false`	

Copyright © 1995-2020 Esri. All rights reserved.

deconflictionStrategy

Optional string that identifies a strategy for avoiding overlap with point symbols or higher priority labels.

Values	Description
`static`	Place the label in the preferred location, unless it would overlap a higher priority label or point feature symbol. If it overlaps a lower priority label, then the lower priority label will disappear or move (depending on whether its deconflictionStrategy is static or dynamic).
`none`	Place the label in the preferred location, regardless of overlaps with other features or labels.
`dynamic`	Place the label in the preferred location, but move to alternative location to avoid overlapping higher priority labels or point feature symbols.
Default = `static`	

labelAngleInfo

Optional object specifying how to position a label following the direction of an angle. These properties will be used if the expression is not empty. The `labelPlacement` will still be used to indicate whether offset or centered positioning is required, but the exact position will be given by the angle calculated for the feature. Once the position has been determined, `textLayout` and `textOrientation` are used to specify the layout of the text at that position.

The `labelAngleInfo` object specifies the angular positions and layout directions for labels on or around point feature symbols. This may be different for each feature (driven by one or more feature attributes) or constant for all features (specified by a fixed number).

Default = `NULL`

Example:

```
"labelAngleInfo": {
  "angleExpressionInfo": {
    "expression": "iif($feature.type == 11, $feature.SITE_ANGLE, 90-2*$feature['other angle'])",
    "returnType": "number"
  },
  "rotationType": "arithmetic"
}
```

labelAngleInfo.angleExpressionInfo

This `expressionInfo` object specifies how the angle (in degrees) for a label is calculated from the feature attributes. It may use attributes, fixed numbers, or a combination of both. If missing, an angle value of zero is assumed.

The object defines a script expression that can be used to compute values. The script may refer to external data which will be available when the expression is being evaluated.

labelAngleInfo.angleExpressionInfo.expression

Optional expression in the Arcade expression language. If no expression is provided, then the default (empty) expression produces a null, empty string, zero or false when evaluated (depending on usage and context). The expression describes how to calculate the rotation angle (in degrees) for a feature, often using attributes of the feature.

Copyright © 1995-2020 Esri. All rights reserved.

labelAngleInfo.angleExpressionInfo.returnType

Always `number` for the `angleExpressioInfo` usage.

labelAngleInfo.rotationType

Optional specification of whether the placement angle calculated by the `angleExpressionInfo` should be interpreted as arithmetic (counter-clockwise from East) or geographic (clockwise from North).

Values	Description
`arithmetic`	Rotation is counter-clockwise from East.
`geographic`	Rotation is clockwise from North.
Default = `arithmetic`	

labelExpression

Optional expression describing how to build the text label for a feature, often using attributes of the feature. More information about the label expression can be found in the REST API Label class topic. If a Webmap or Arcade expression (`labelExpressionInfo`) is also provided, `labelExpression` is ignored.

Default = `""` (an empty expression that results in no labels being generated)

Expression examples:
- "You are here"—Fixed text is used as the label for every feature.
- "[AreaName]"—The value of the `AreaName` attribute for each feature is used as the label.
- "[StreetName] CONCAT " Street""—Append the `StreetName` attribute value with the text "Street" for each feature's label.

labelExpressionInfo

An optional object containing an Arcade or Webmap expression that describes how to label features.

labelExpressionInfo.expression

An optional Arcade expression describing how to build the text label for a feature, often using attributes of the feature. More information about building Arcade expressions can be found in the ArcGIS Arcade documentation.

Default = `""` (an empty expression that results in no labels being generated)

Expression examples:
- ""You are here""—Fixed text is used as the label for every feature.
- "$feature.AreaName"—The value of the `AreaName` attribute for each feature is used as the label.
- "$feature.StreetName + ' Street'"—Append the `StreetName` attribute value with the text "Street" for each feature's label.

labelExpressionInfo.value

An optional Webmap expression describing how to build the text label for a feature, often using attributes of the feature. This language allows embedding of the feature's attributes in a literal text string. If an Arcade expression (`labelExpressionInfo.expression`) is also provided, this expression is ignored.

Copyright © 1995-2020 Esri. All rights reserved.

Default = `""` (an empty expression that results in no labels being generated)

Expression examples:

- "You are here"—Fixed text is used as the label for every feature.
- "{AreaName}"—The value of the `AreaName` attribute for each feature is used as the label.
- "{StreetName} Street"—Append the `StreetName` attribute value with the text "Street" for each feature's label.

labelPlacement

An optional preferred position of the text label, with respect to its feature geometry.

Values	Description
`esriServerPointLabelPlacementAboveCenter`	Lower midpoint of label is offset North of point symbol.
`esriServerPointLabelPlacementAboveLeft`	Lower right corner of the label is offset North-west of point symbol.
`esriServerPointLabelPlacementAboveRight`	Lower left corner of the label is offset North-east of point symbol.
`esriServerPointLabelPlacementBelowCenter`	Upper midpoint of label is offset South of point symbol.
`esriServerPointLabelPlacementBelowLeft`	Upper right corner of the label is offset South-west of point symbol.
`esriServerPointLabelPlacementBelowRight`	Upper left corner of the label is offset South-east of point symbol.
`esriServerPointLabelPlacementCenterCenter`	Center of label is placed on geometry point.
`esriServerPointLabelPlacementCenterLeft`	Right midpoint of label is offset West of point symbol.
`esriServerPointLabelPlacementCenterRight`	Left midpoint of label is offset East of point symbol.
`esriServerLinePlacementAboveAfter`	Lower left corner of label is at final geometry coord, label extrapolates the last geometry segment.
`esriServerLinePlacementAboveAlong`	Lower midpoint of label prefers the midpoint of the geometry, label follows the geometry segments.
`esriServerLinePlacementAboveBefore`	Lower right corner of label is at first geometry coord, label extrapolates the first geometry segment.
`esriServerLinePlacementAboveStart`	Lower left corner of label is at first geometry coord, label follows the first geometry segments.
`esriServerLinePlacementAboveEnd`	Lower right corner of label is at final geometry coord, label follows the last geometry segments.
`esriServerLinePlacementBelowAfter`	Upper left corner of label is at final geometry coord, label extrapolates the last geometry segment.
`esriServerLinePlacementBelowAlong`	Upper midpoint of label prefers the midpoint of the geometry, label follows the geometry segments.
`esriServerLinePlacementBelowBefore`	Upper right corner of label is at first geometry coord, label extrapolates the first geometry segment.
`esriServerLinePlacementBelowStart`	Upper left corner of label is at first geometry coord, label follows the first geometry segments.
`esriServerLinePlacementBelowEnd`	Upper right corner of label is at final geometry coord, label follows the last geometry segments.

Copyright © 1995-2020 Esri. All rights reserved.

Values	Description
`esriServerLinePlacementCenterAfter`	Left midpoint of label is at final geometry coord, label extrapolates the last geometry segment.
`esriServerLinePlacementCenterAlong`	Center of label prefers the midpoint of the geometry, label follows the geometry segments.
`esriServerLinePlacementCenterBefore`	Right midpoint of label is at first geometry coord, label extrapolates the first geometry segment.
`esriServerLinePlacementCenterStart`	Left midpoint of label is at first geometry coord, label follows the first geometry segments.
`esriServerLinePlacementCenterEnd`	Right midpoint of label is at final geometry coord, label follows the last geometry segments.
`esriServerPolygonPlacementAlwaysHorizontal`	Center of label is as far inside polygon as possible.

Default depends on the geometry type of the feature:

- Points = `esriServerPointLabelPlacementAboveRight`
- Lines = `esriServerLinePlacementAboveAlong`
- Polygons = `esriServerPolygonPlacementAlwaysHorizontal`

lineConnection

Optional string to control whether line features with the same label, and matching end vertices, should be joined before sharing a label.

Values	Description
`minimizeLabels`	Line geometries with the same label and coincident end vertices should be considered as a single geometry when placing labels.
`unambiguousLabels`	Line geometries with the same label and coincident end vertices should be considered as a single geometry when placing labels, until they hit a junction.
`none`	Keep one label per line feature geometry.

Default = `minimizeLabels`

maxScale

Limit the visibility of the labels to scales smaller than the value provided.

Values	Description
`0`	No maximum scale.
`Number`	Denominator of the largest scale at which labels are still visible.

Default = `0`

minScale

Limit the visibility of the labels to scales larger than the value provided.

Copyright © 1995-2020 Esri. All rights reserved.

Values	Description
`0`	No minimum scale.
`Number`	Denominator of the smallest scale at which labels are still visible.
Default = `0`	

multiPart

How to handle labels for multipart polyline or polygon features.

Values	Description
`labelLargest`	If a line feature consists of multiple line geometries, or a polygon feature consists of multiple ring geometries, the largest one will be assigned a label.
`labelPerPart`	If a line feature consists of multiple line geometries, or a polygon feature consists of multiple ring geometries, each geometry will be assigned a label.
`labelPerSegment`	Not supported.
`labelPerFeature`	Not supported.
Default = `labelPerPart`	

name

A text string to identify the label class, which may be presented in user interfaces where users interact with label classes.

Default = `""` (no name for the label class)

offsetDistance

Optional specification of the screen distance (in points) between the feature symbol geometry and an offset label. This can be a negative value to pull the label closer to the feature.

Default = `1`

priority

A number that controls which labels are placed first and can also supplant existing lower-priority labels.

Values	Description
`-1`	Automatic.
`0`	Most important.
`5`	High priority.
`15`	Moderate priority.
`25`	Low priority.
`Higher values`	Lowest priority.

Copyright © 1995-2020 Esri. All rights reserved.

Values	Description
Default = `-1` The actual priority is calculated based on the geometry type: • Point features = `12` • Line features = `15` • Polygon features = `18`	

removeDuplicates

An optional setting to control how features with the same text are labeled. The distance that defines which labels are considered "nearby" is specified with the **removeDuplicatesDistance** setting.

Values	Description
`none`	No duplicate labels should be removed. Leave all the placed labels on the map regardless of nearby duplicates.
`labelClass`	Remove nearby duplicate labels if they belong to the same Label Class.
`featureType`	Remove nearby duplicate labels if they belong to the same Feature Type (point vs line vs polygon).
`all`	Remove nearby duplicate labels.
Default = `none`	

removeDuplicatesDistance

Specifies the duplicate thinning radius (in points) to use with the **removeDuplicates** setting.

Values	Description
`0`	Special indicator value meaning infinity (the entire extent, in other words).
`Number`	Duplicate thinning radius distance in points.
Default = `0`	

repeatLabel

For line features, whether or not multiple copies of the label be placed for each feature.

Values	Description
`true`	Repeat the label along the feature.
`false`	Only place one label per feature.
Default = `true` for line geometry, `false` for points and polygons.	

repeatLabelDistance

Indicate how dense the repeated labels are (in points). The distance is considered a guideline, the exact placement will be affected by the geometry and other nearby features and labels.

Copyright © 1995-2020 Esri. All rights reserved.

Values	Description
`Number`	Distance along the line feature in points, between the end of one repetition and the start of the next.
Default = `216` (in points, approximately 3 inches)	

stackAlignment

Control the alignment of stacked label text.

Values	Description
`dynamic`	Set the alignment depending upon the label's position with respect to its feature. For example, a stack to the right of a point symbol will be left-aligned.
`textSymbol`	Follow the alignment specified in the text-symbol.
Default = `textSymbol`	

stackBreakPosition

Controls whether a row of text should be broken before or after it exceeds the ideal length. If stacking is turned on, a line break can be inserted before or after the breaking word that overruns the maximum number of characters per row. Using the `before` option means rows will generally be shorter than the `stackRowLength` although they will overrun for individual words larger than this count.

Values	Description
`before`	Insert the line break before a word that will break the `stackRowLength` limit.
`after`	Insert the line break after a word that has broken the `stackRowLength` limit.
Default = `after`	

stackLabel

Whether text should be stacked or wrapped, rather than placing long labels across the map.

Values	Description
`true`	Break long labels (more than 9 characters) and insert line breaks between words.
`false`	Do not change the label. If the label contains line breaks already, they will be honored.
Default = `true` for point and polygon features, `false` for line features	

stackRowLength

The character limit for a row of text. A line break is inserted for the label when this limit is reached.

Values	Description
`-1`	Use default length.
`0 or higher`	Insert a line break when a row exceeds this number of characters.
Default = `-1`	

Copyright © 1995-2020 Esri. All rights reserved.

stackSeparators

Optional specification of the separators that should be used for line breaks. These are defined with a list of `LabelStackSeparator` objects with the following properties:

Values	Description
`separator`	One character codepoint to look for in text string.
`visible`	Whether or not the separator character should be displayed if chosen as a line break position (`true` or `false`, default = `false`).
`forced`	Whether or not the separator should always be used as a line break position (`true` or `false`, default = `false`).
`breakPosition`	Break a line of text before or after it exceeds the ideal length (`before` or `after`, default = `after`).
Default separators = single space, comma, and hyphen	

Example:

```
"stackSeparators": [{separator:" ",visible:true,forced:false,breakPosition:"after"},
                    {separator:";",visible:false,forced:false,breakPosition:"after"}]
```

symbol

The text symbol to use for displaying labels. In addition to familiar properties such as color and font, the text symbol can define things like offsets, alignment, rotation, and angle.

Values	Description
`angle`	A numeric value that defines the number of degrees (0 to 360) that a text symbol is rotated. The rotation is from East in a counter-clockwise direction where East is the 0° axis.
`backgroundColor`	Background color is represented as a four-element array. The four elements represent values for red, green, blue, and alpha in that order. Values range from 0 to 255. If color is undefined, the color value is null.
`borderLineColor`	Borderline color is represented as a four-element array. The four elements represent values for red, green, blue, and alpha in that order. Values range from 0 to 255. If color is undefined, the color value is null.
`borderLineSize`	The width of the border, in points.
`color`	Text color is represented as a four-element array. The four elements represent values for red, green, blue, and alpha in that order. Values range from 0 to 255. If color is undefined, the color value is null.
`font`	A `font` object that specifies the font used for the symbol. `Font` defines `decoration`, `family`, `size`, `style`, and `weight` properties.
`haloColor`	A color to display around the text as a halo. If color is undefined, the color value is null.
`haloSize`	The size of a halo to draw around the text, in points.
`horizontalAlignment`	One of the following values to define horizontal alignment of the text: `left`, `right`, `center`, `justify`.
`kerning`	A boolean to specify whether or not to adjust spacing between characters in the text.

Copyright © 1995-2020 Esri. All rights reserved.

Values	Description
`rightToLeft`	A boolean to specify whether or not to render text from right to left (for Arabic or Hebrew text, for example).
`rotated`	A boolean to specify whether or not every character in the text is rotated.
`text`	Text to render with the symbol. This is only applicable when specified as a symbol for a client-side graphic.
`type`	The type of symbol. Must be `esriTS` for a text symbol.
`verticalAlignmet`	One of the following values to define vertical alignment of the text: `top`, `bottom`, `middle`, `baseline`.
`xoffset`	The distance to offset the symbol along the x-axis, in points.
`yoffset`	The distance to offset the symbol along the y-axis, in points.

Example:

```
{
  "symbol": {
    "angle": 0,
    "backgroundColor": [
      0,
      0,
      0,
      0
    ],
    "borderLineColor": [
      255,
      0,
      255,
      255
    ],
    "borderLineSize": 2,
    "color": [
      78,
      78,
      78,
      255
    ],
    "font": {
      "decoration": "none",
      "family": "Arial",
      "size": 12,
      "style": "normal",
      "weight": "bold"
    },
    "haloColor": [
      0,
      255,
      0,
      255
    ],
    "haloSize": 2,
    "horizontalAlignment": "left",
    "kerning": true,
    "rightToLeft": false,
    "type": "esriTS",
    "verticalAlignment": "bottom",
    "xoffset": 0,
```

Copyright © 1995-2020 Esri. All rights reserved.

```
    "yoffset": 0
  }
}
```

textLayout

Optional specification of whether the label text should be written horizontally, straight in line with the (line) feature or (point) positioning angle, or perpendicular to the positioning angle.

Values	Description
`automatic`	Varies by geometry type. See the description of default behavior.
`horizontal`	Layout the text horizontally.
`straight`	Layout the text straight in line with the (line) feature or (point) positioning angle.
`perpendicular`	Layout the text perpendicular to the positioning angle.
Default = `automatic`, which results in: • `horizontal` for point feature labels. • `straight` for line feature labels. • `horizontal` for polygon feature labels.	

textOrientation

Optional specification of whether text should follow the placement angle direction even if it means being rendered upside-down, or whether text should be flipped 180 degrees to keep it page-oriented.

Values	Description
`direction`	Text should follow the placement angle direction even if it means being rendered upside-down.
`page`	Text should be flipped 180 degrees to keep it page-oriented.
Default = `page`	

useCodedValues

A boolean that controls whether the data source should translate domain identifiers into descriptions. This is only used for simple expressions, since it is ignored if an Arcade or Webmap expression is used. Arcade provides its own function for the user to specify decoding on demand.

Values	Description
`true`	Attributes are translated if they have domain mappings set up in the datasource.
`false`	Raw attributes are used.
Default = `false`	

where

An expression to limit the visibility of labels in the label class.

Copyright © 1995-2020 Esri. All rights reserved.

Examples:

- "CITY_NAME LIKE 'San%'"
- "(POP2000 > 2000000) AND (CAPITAL = 'Y')"

Default = `""` (empty string which means all labels in the class are displayed)

Copyright © 1995-2020 Esri. All rights reserved.

Support for the web scene specification

ArcGIS Runtime SDK provides partial read support for the Esri web scene specification. This page describes support for web scenes and associates web scene features with classes.

This release of ArcGIS Runtime does not support persistence (write, edit, save or upload) of web scenes.

Scenes

ArcGIS Runtime supports basic features of global scenes, including basemaps, ground surfaces, operational layers, initial viewpoints, tables, and spatial references. It does not support local scenes, environment (lighting) settings, app metadata, presentation slides, clipping, range info, or vertical coordinate systems.

Web scene field	ArcGIS Runtime support
baseMap (except id and transparency)	Scene.Basemap
ground	Scene.BaseSurface
initialState.viewpoint	Scene.InitialViewpoint
operationalLayers	Scene.OperationalLayers
spatialReference	Scene.SpatialReference
tables	Scene.Tables

ArcGIS Runtime supports only scenes with the following values set:

- **version** - 1.10.0 or greater
- **viewingMode** - global

Layers

ArcGIS Runtime supports a subset of the layers supported by web scenes. The following table associates web scene layers with ArcGIS Runtime layer types.

Web scene layer	ArcGIS Runtime layer type
ArcGISFeatureLayer	FeatureLayer
ArcGISImageServiceLayer	ArcGISMapImageLayer
ArcGISMapServiceLayer	ArcGISMapImageLayer
IntegratedMeshLayer	ArcGISSceneLayer
OpenStreetMap	OpenStreetMapLayer
ArcGISSceneServiceLayer	ArcGISSceneLayer
ArcGISTiledMapServiceLayer	ArcGISTiledLayer
WebTiledLayer	WebTiledLayer
WMS	WMSLayer

The following layers are either not supported or not supported in scenes:

- CSV layer

Copyright © 1995-2020 Esri. All rights reserved.

- VectorTileLayer - supported but will not display in scenes

Renderers

ArcGIS Runtime only supports a subset of renderers consumed through web scenes.

Web scene renderer type	Runtime renderer type
ClassBreaks	ClassBreaksRenderer
Simple	SimpleRenderer
UniqueValue	UniqueValueRenderer

Visual variables are not supported on any renderers.

The following renderers supported by the web scene specification are not supported by ArcGIS Runtime:

- PointCloud ClassBreaks renderer
- PointCloud Stretch renderer
- PointCloud UniqueValue renderer
- Raster ClassBreaks renderer
- Raster Stretch renderer
- Raster UniqueValue renderer
- UniqueValueFromStyle renderer

Symbols

The following table associates supported web scene symbol types with ArcGIS Runtime SDK symbol types:

Web scene symbol type	ArcGIS Runtime symbol type
LineSymbol3D	MultilayerPolylineSymbol
PointSymbol3D	MultilayerPointSymbol
PolygonSymbol3D	MultilayerPolygonSymbol
StyleSymbolReference	MultilayerPointSymbol

Symbol layers

ArcGIS Runtime supports reading and rendering symbol layers. However, there are no API types available for manipulating these layers:

- ObjectSymbol3DLayer

The following symbol layers are not supported:

- ExtrudeSymbol3DLayer

Elevation sources

Runtime supports the ArcGISTiledElevationServiceLayer from the web scene spec, which is exposed as ArcGISTiledElevationSource.

The following table associates elevation source properties with the equivalent ArcGIS Runtime types:

Copyright © 1995-2020 Esri. All rights reserved.

Web scene property	ArcGIS Runtime property
itemId	ArcGISTiledElevationSource.Item
layerType	N/A - only ArcGISTiledElevationSource is supported
title	ArcGISTiledElevationSource.Name
url	ArcGISTiledElevationSource.Source
visibility	ArcGISTiledElevationSource.IsEnabled

Copyright © 1995-2020 Esri. All rights reserved.

Legal

Legal notices are installed in the `legal` subfolder of your project's ArcGIS Runtime SDK for .NET NuGet package folder and include the following:

- Copyright and Trademarks
- Master Agreement - Products and Services

For the latest versions of these documents and several others, see http://www.esri.com/legal/.

Copyright © 1995-2020 Esri. All rights reserved.