esri

Esri International Developer Summit

Palm Springs, CA

Migrating your WPF Apps to the New ArcGIS Runtime SDK for .NET

Mike Branscomb

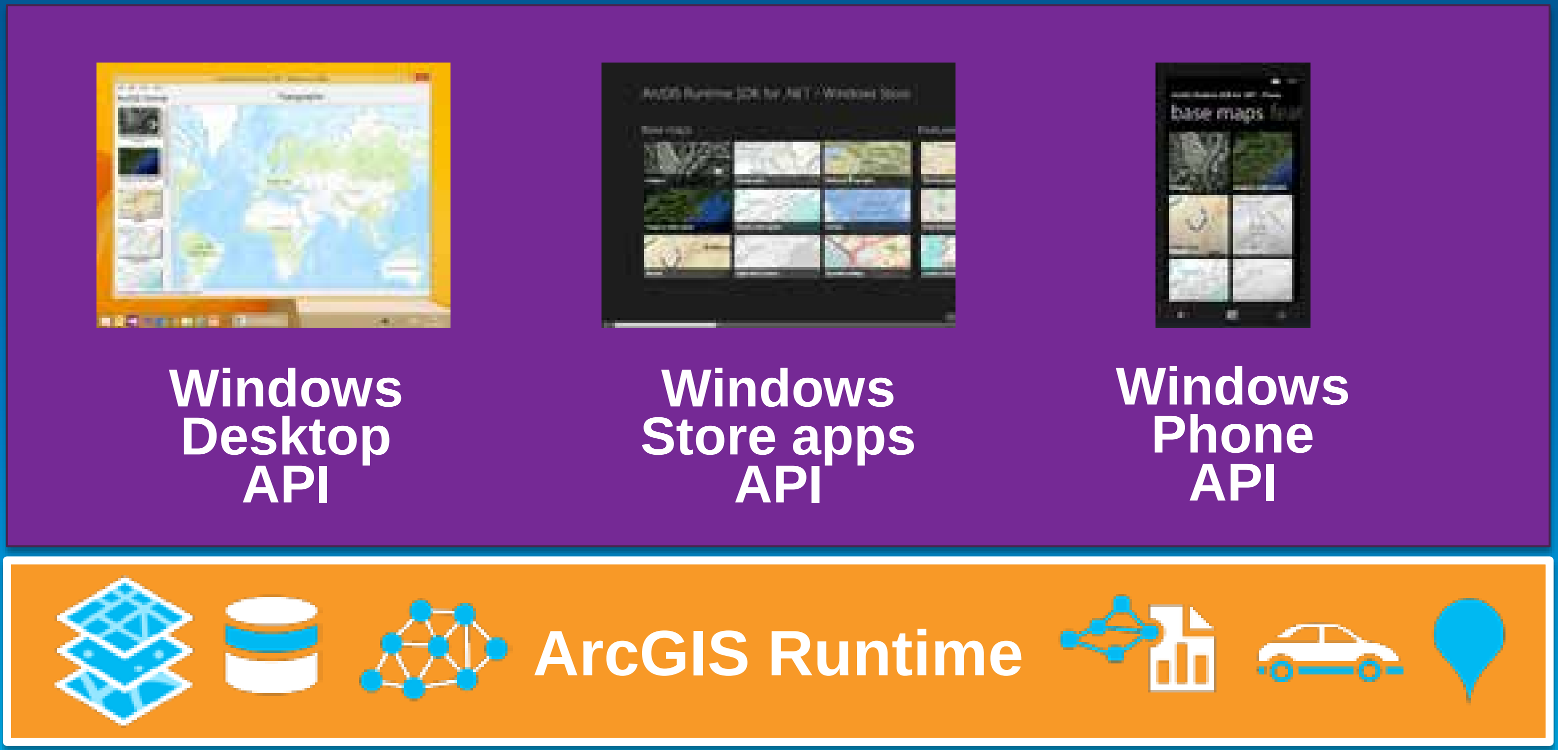
Antti Kajanus

Agenda

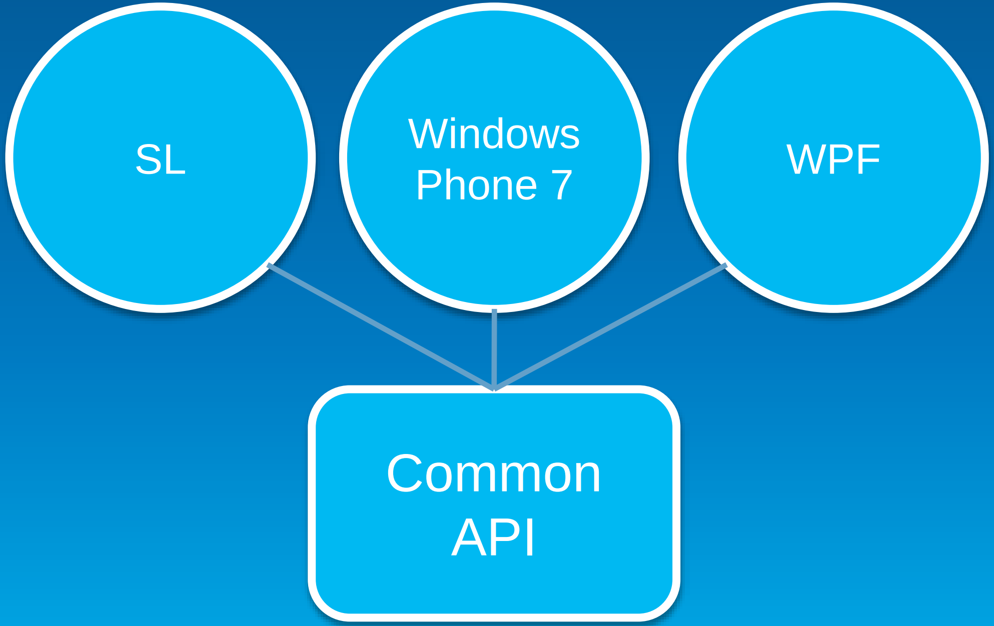
- Comparison of WPF SDK and .NET SDK
- Windows Desktop API
- Do you need to migrate?
- Preparing to migrate
- Migrating a WPF app
- Differences between the APIs

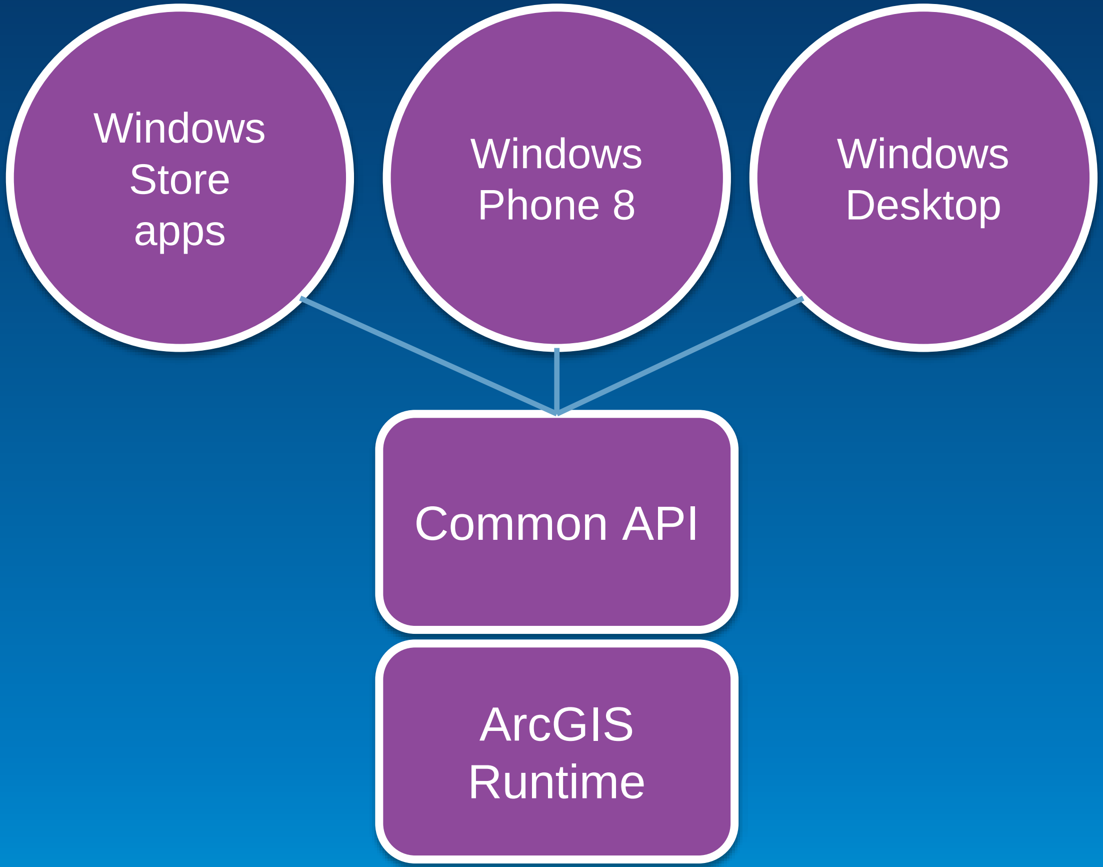
ArcGIS Runtime SDK for .NET

- Supports .NET development on three platforms
  - Desktop, Store apps, and Phone
- Shared API design and functionality
- Built on ArcGIS Runtime

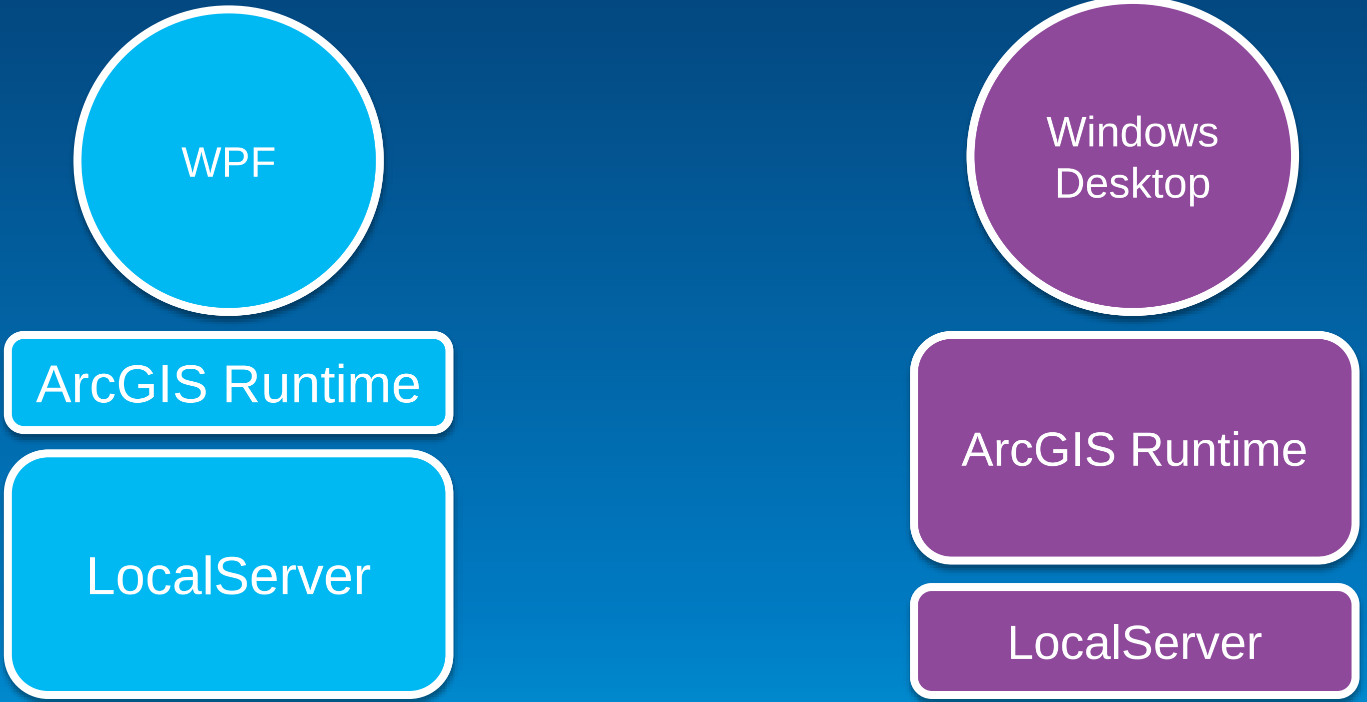


WPF to Win Desktop





WPF to Win Desktop



WPF vs. Win Desktop

WPF SDK	.NET SDK – Windows Desktop
WPF	WPF
.NET 4.0	.NET 4.5
SL / WPF / Phone 7	Desktop / Store / Phone 8
Services-based + ArcGIS Runtime / LocalServer	ArcGIS Runtime-based + LocalServer
Event-based Async Pattern (EAP)	Task-based Async Pattern (TAP)
Some MVVM support	Designed for MVVM

Preparing for the new .NET SDK

- What about all the great skills and knowledge you have from using the existing WPF SDK?
  - All still very relevant for the new .NET SDK
  - Many shared concepts, class names and class members
  - But new API is:
    - Based on .NET 4.5
    - Uses async Tasks instead of events
    - Designed for MVVM
    - Refined for consistency within the API and with other ArcGIS Runtime APIs
    - Built with the benefit of hindsight

http://blogs.esri.com/esri/arcgis/2014/03/07/getting-ready-for-the-new-net-sdk/

Preparing for the new .NET SDK

- Do I need to migrate all my WPF apps?
  - Perhaps not…
  - Transition will involve some redevelopment
  - Review on an app-by-app basis
  - Does your app need functionality in the new API?
  - If no then continue to build with WPF SDK
  - If yes then plan for migration…
  - Start now by taking advantage of 10.2 / 10.2.2 and .NET 4.5 / C# 5.0
    - Async Tasks

DEMO

WPF App 10.1.1-style

Antti Kajanus

Demo Summary - 10.1.1 app

- App buffers user click point, performs spatial query and displays States
- Starts LocalGeometryService
  - StartAsync / StartCompleted inline Lambda expression
- LocalMapService
  - GetServiceAsync Action delegate inline event handling
- Map.MouseClick
  - ArcGIS API for WPF event
  - Returns MapPoint
- GeometryService task
  - BufferCompleted and Failed events
- QueryTask
  - Queries LocalMapService
  - ExecuteCompleted and Failed events

Preparing for the new .NET SDK

Tip 1/3: Use the accelerated display

- ArcGIS Runtime GIS-optimized rendering pipeline
- Same rendering used by all ArcGIS Runtime SDKs
- Particularly beneficial for graphics and features
- Enabled via Map.UseAcceleratedDisplay

```
<esri:Map x:Name="MyMap" UseAcceleratedDisplay="True" WrapAround="True">
    <esri:ArcGISTiledMapServiceLayer ID="MyLayer"
        Url="http://services.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer"/>
    <esri:GraphicsLayer ID="MyGraphics" Renderer="{StaticResource MySimpleRenderer}"/>
</esri:Map>
```

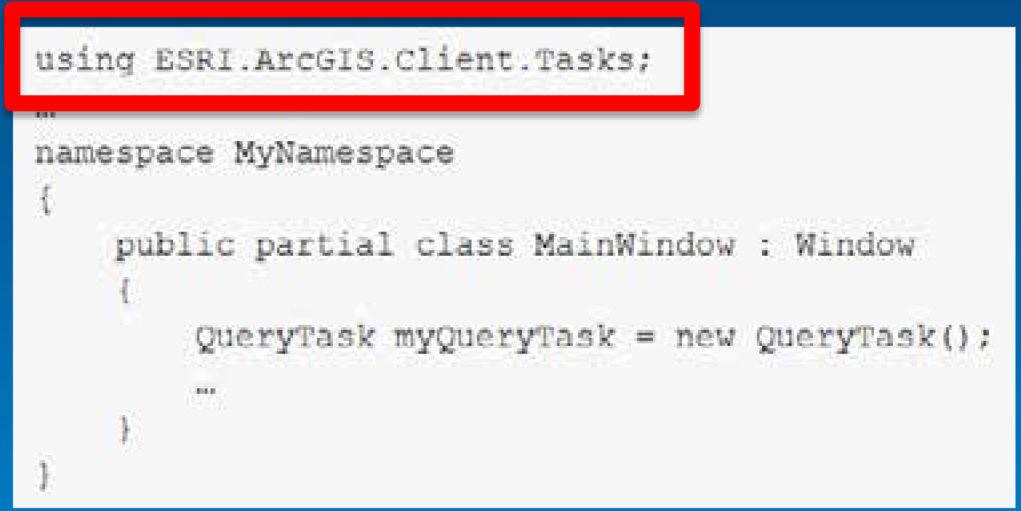
- Esri symbol and renderer types only
  - No custom or animated symbols
- Some Layer types not supported
  - ElementLayer, KmlLayer

Tip 2/3: Use Async Tasks

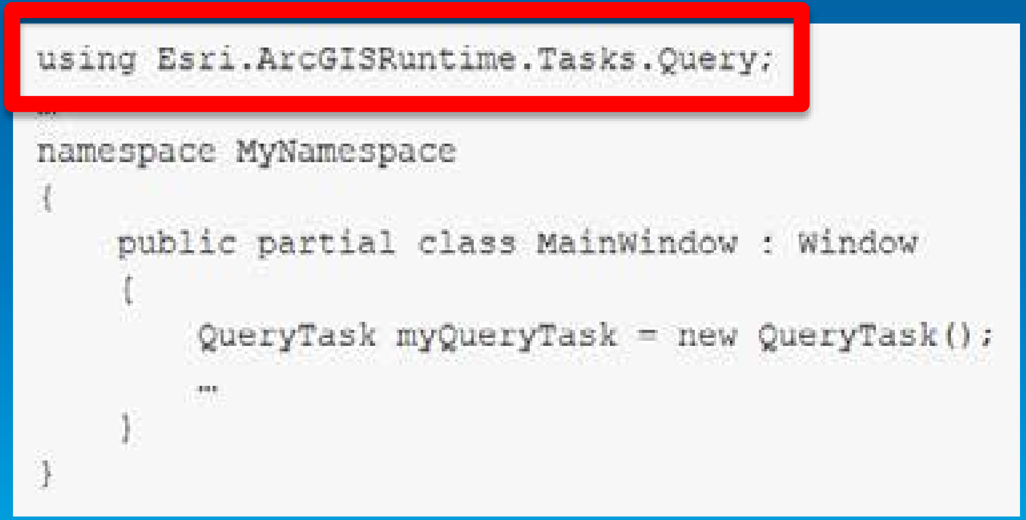
- Tasks arrived with .NET 4.0
- Simplified async programming
- No need to manage threads or use BackgroundWorker
- Became easy to use with .NET 4.5 & C# 5.0 (VS2012)
  - ‘async and await’ keywords
- 10.2 added overloads to existing async methods
  - Return async Tasks
- 10.2 added some new methods
  - e.g. ExecuteAsync + ExecuteTaskAsync

Tip 3/3: Use using statements for namespaces

- Do not fully qualify classes
- Use the using statement to reference namespaces



```
using ESRI.ArcGIS.Client.Tasks;
…
namespace MyNamespace
{
    public partial class MainWindow : Window
    {
        QueryTask myQueryTask = new QueryTask();
        …
    }
}
```



```
using Esri.ArcGISRuntime.Tasks.Query;
…
namespace MyNamespace
{
    public partial class MainWindow : Window
    {
        QueryTask myQueryTask = new QueryTask();
        …
    }
}
```

DEMO

Taking advantage of 10.2

Antti Kajanus

Demo Summary - 10.2 app

- App buffers user click point, performs spatial query and displays States
- .NET 4.5
- Starts LocalGeometryService
  - StartAsync awaitable Task
- LocalMapService
  - GetServiceAsync awaitable Task
- Map.MouseClick
  - Custom event returns MapPoint
- GeometryService task
  - BufferTaskAsync awaitable Task
- QueryTask
  - Queries LocalMapService
  - ExecuteTaskAsync awaitable Task

Simplified async code

Migrating to Windows Desktop API

DEMO

Migrating to Windows Desktop API

Antti Kajanus

Demo Summary – Windows Desktop app

- App buffers user click point, performs spatial query and displays States
- New assembly and namespace Esri.ArcGISRuntime…
- Map changed to MapView / Map
- Uses GeometryEngine
  - No LocalGeometryService or GeometryService task
- Uses GeodatabaseFeatureTable
  - Does not need LocalMapService for QueryTask
- Uses MapView.MapViewTapped event
  - No Map.MouseClick event
- Queries GeodatabaseFeatureTable directly
  - Does not need to use QueryTask

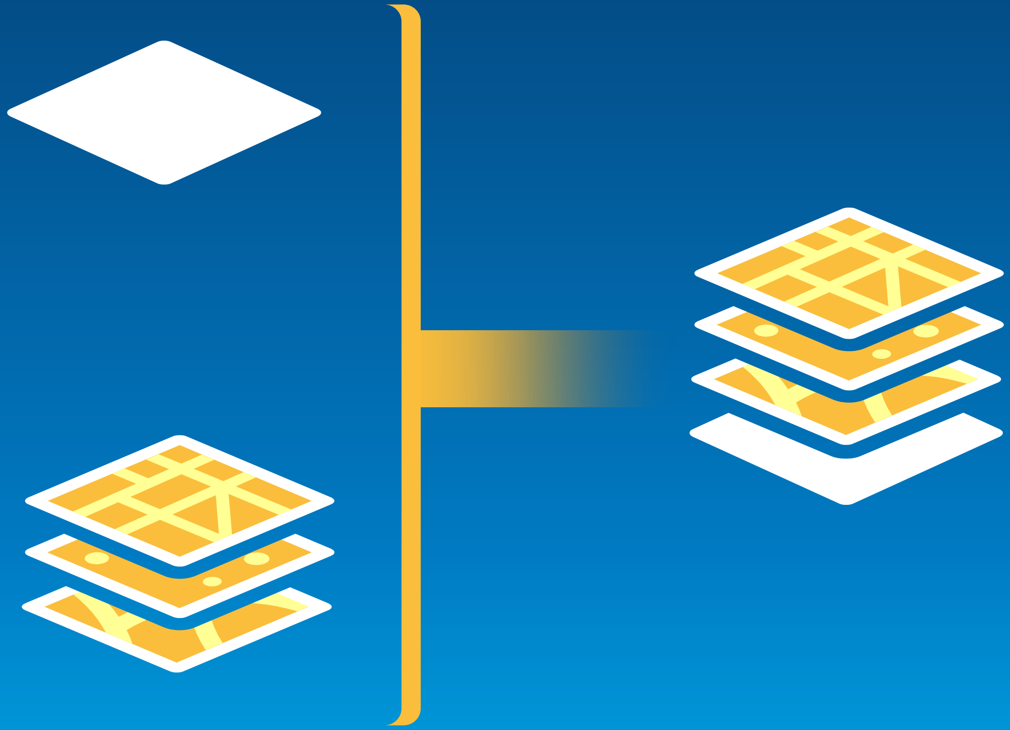
API Comparison

The Map

- WPF Map split into:

- MapView
  - UI container for a Map
  - Display related properties
    - WrapAround
    - SpatialReference
    - LocationDisplay

- Map
  - Object with a collection of layers
  - InitialExtent
  - Bindable to support MVVM



The Map

- No Map.UseAcceleratedDisplay property
  - Map is always ‘accelerated’
- No settable Map.Extent property
  - Use Map.InitialExtent
- No PanTo method
  - Use MapView.SetView / SetViewAsync
- No ZoomTo method
  - Use ZoomAsync / ZoomToScaleAsync

The Map

- Map events are now MapView events
- No ArcGIS API for WPF ‘MapClicked’ event
- New unified interaction model for mouse / stylus / touch
  - MapView.MapViewTapped
  - MapView.MapViewDoubleTapped
  - MapView.MapViewHolding
- Avoids need for separate events based on interaction mode

Async Tasks

- New API uses Task-based async pattern
- Replaces event-based async pattern
- Greatly simplifies async code
- Recommend using ‘await’ keyword to make async calls
- Use Tasks instead of BackgroundWorker for threads
- Tasks raise exceptions
  - Instead of ‘Failed’ style events
- Use Try-Catch appropriately to handle exceptions

Layers

- New layer initialization pattern
  - No Layer Initialized / InitializationFailed events
  - New Layer.InitializeAsync awaitable Task
  - New MapView LayerLoaded / LayerUnloaded events
  - New MapView.LayersLoadedAsync awaitable Task
    - E.g. await MyMapView.LayersLoadedAsync();

- No Layer.Url property
  - Use Layer.ServiceUri

- No Local Dynamic / Local Feature Layer types
  - Use online service types and set ServiceUri property once service has started

FeatureLayer

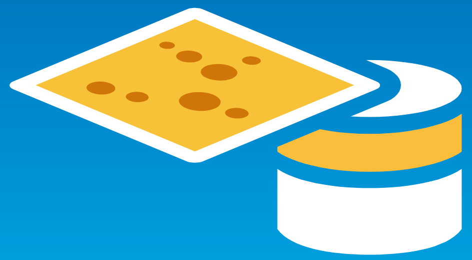
- Feature data pattern is very different
- From FeatureTables
  - GeodatabaseFeatureTable
  - GeodatabseFeatureServiceTable
- FeatureLayer no longer derives from GraphicsLayer
- Features do not have display-related properties
- No RenderingMode property
  - Always uses ‘Static’ mode
- Can get graphics from features
  - Feature.AsGraphic()
- Query via direct API on to FeatureTable
  - No need to use QueryTask and make service call

Symbols and Renderers

- No custom XAML symbols
- Esri symbol types only
  - Simple: SimpleMarkerSymbol, SimpleLineSymbol, SimpleFillSymbol
  - Picture: PictureMarkerSymbol, PictureFillSymbol
  - CompositeSymbol
  - TextSymbol
- Can use PictureMarkerSymbol and CompositeSymbol to replicate some XAML symbols or try RenderTargetBitmap approach
- Symbol colors based on Color instead of SolidColorBrush
- FeatureLayers only have renderers
  - No Feature Symbol property
- No MapTips
  - Use MapOverlay

Editing

- Draw class replaced by Editor class
  - No Draw.Enabled / DrawComplete
- Use the Editor for all editing and to capture user interaction as geometries
- New awaitable Task Editor.RequestShapeAsync / EditShapeAsync
- No out of the box EditorWidget
  - Should we add this…?
- Need to programmatically create / edit features and attributes
- This is straightforward with simpler API



Geometry

- Same standard geometry types
  - MapPoint, Polyline, Polygon, Envelope, etc
- Plus simple lightweight geometry representations
  - Coordinate
  - CoordinateCollection
    - E.g. Polyline.Paths
    - E.g. Polygons.Rings
- Any use of PointCollection will need to change to CoordinateCollection

Geometry

- GeometryService task replaced by GeometryEngine
  - Almost identical list of operations: Buffer, Project, etc
- ArcGIS Runtime based geometry operations
- No service calls
- Synchronous operations
- Very fast

LocalServer

- Windows Desktop API includes LocalServer
- Supports existing Package-based workflows
- API still includes:
  - LocalMapService
  - LocalFeatureService
  - LocalGeoprocessingService
- No Local layer types
  - Use online layer types with the Url of the service
  - Need to manage service lifetime in code
- LocalLocatorTask replaces LocalGeocodeService
- LocalGeometryEngine replaces LocalGeometryService
- LocalRouteTask replaces GP for routing

Package-based workflows?

- Many Map Package based workflows will benefit from new runtime geodatabase
- Where you previously used:
  - LocalMapService / LocaFeatureService
  - ArcGISLocalDynamicMapServiceLayer / ArcGISLocalFeatureLayer
- Instead use:
  - Extract runtime gdb from feature service (and sync)
  - Export runtime gdb from ArcMap (read-only in 10.2.2)
  - Access local geodatabase feature tables directly
  - Add to Map as FeatureLayer
  - Display, query, edit, sync

DEMO

Why migrate?

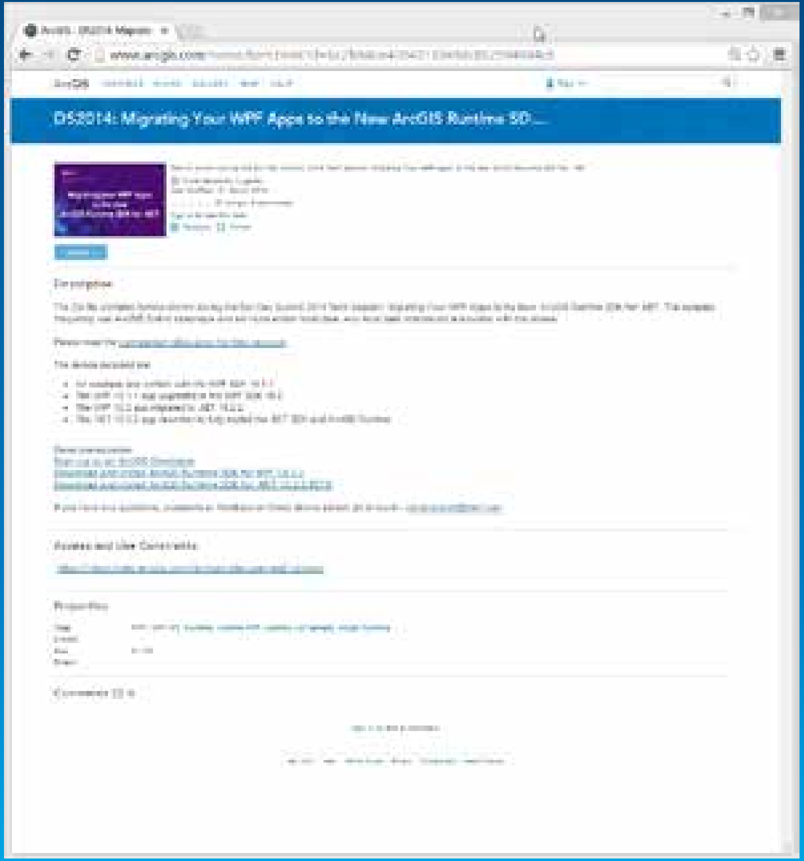
Antti Kajanus

Summary

- Review your app
  - Does it need new functionality?
  - Can it keep building on WPF SDK?
- Take advantage of 10.2 to prepare your app for migration
  - Use accelerated display
  - Async Tasks
  - Use using statements to import namespaces
- Recommend using these features even if you do not plan to migrate!
- .NET SDK offers many advantages…

Demos available on ArcGIS.com

http://esriurl.com/7575



www.arcgis.com[illegible]

DS2014: Migrating Your WPF Apps to the New ArcGIS Runtime SD...

Description

[illegible]

Access and Use Constraints

[illegible]

Properties

[illegible]

esri

Understanding our world.