ArcGIS Pro SDK for .NET: Beginning Pro Customization and Extensibility

- Uma Harano
- Wolf Kaiser

2019 ESRI DEVELOPER SUMMIT
Palm Springs, CA

Session Overview

- The Pro SDK supports Two Main Extensibility patterns
  - ArcGIS Pro Module Add-ins
  - ArcGIS Pro Managed Configurations
- Declarative Markup for Pro UI Elements: DAML
  - How to change the ArcGIS Pro UI with your Add-in
- Asynchronous Programming
  - Authoring your own asynchronous functions
  - Use of async and await keywords, introduction to QueuedTask

What is the ArcGIS Pro SDK for .NET?

- ArcGIS Pro SDK for .NET extends ArcGIS Pro using .NET
- To implement extensibility patterns the Pro SDK provides Visual Studio project templates:
  - ArcGIS Pro Module Add-ins
  - ArcGIS Pro Managed Configurations
  - ArcGIS Pro Plug-in Datasource (new in Pro 2.3)
- Includes Visual Studio item templates for most UI elements
- Installation is integrated with Visual Studio Marketplace
- ArcGIS Pro API comes with ArcGIS Pro out-of-box
  - File based references (No Global Assembly Cache)

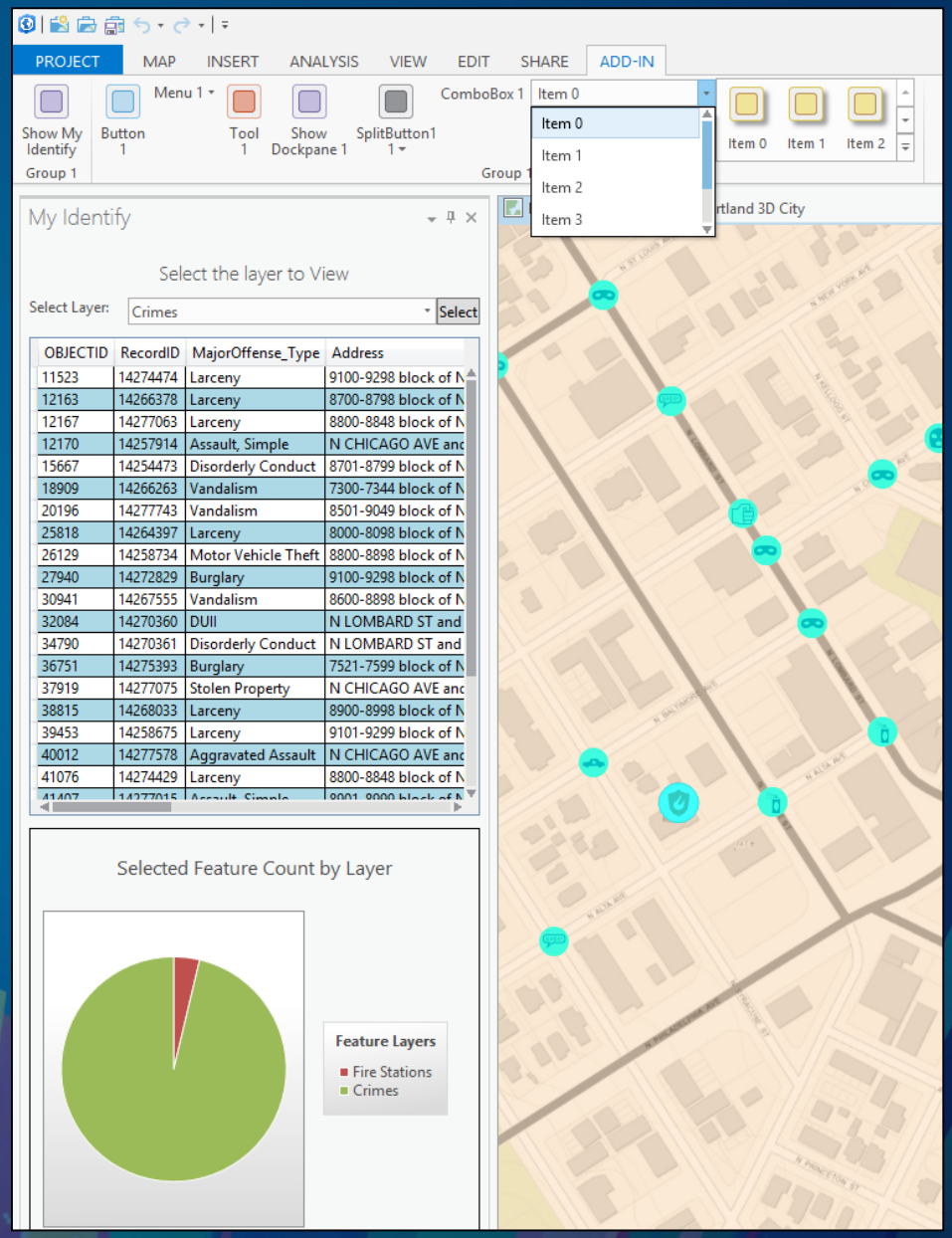
What is the ArcGIS Pro SDK for .NET?

(Continued)

- Noteworthy ArcGIS Pro SDK features and patterns
  - 64-bit platform
  - UI based on WPF (Windows Presentation Foundation) / .NET 4.6.1 +
  - MVVM pattern (Model-View-ViewModel)
  - Asynchronous Patterns: Multiple threads

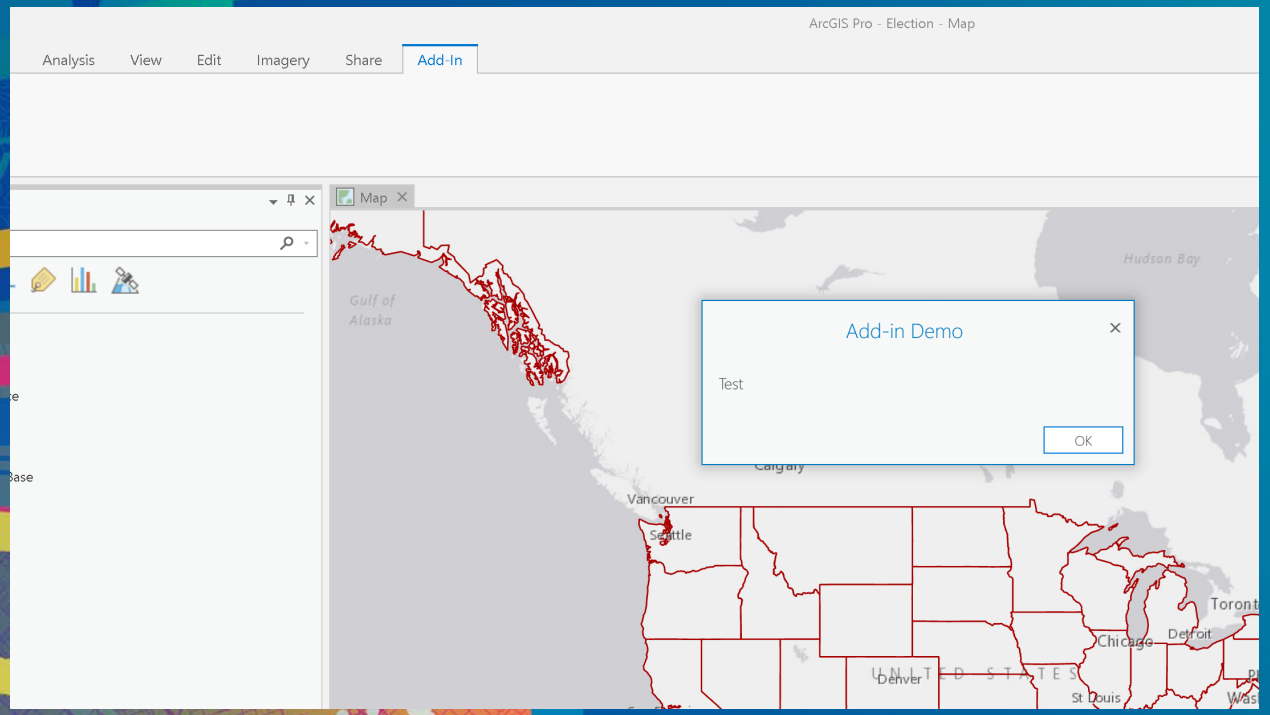
What is an ArcGIS Pro add-in ?

- Extends ArcGIS Pro through:
  - Buttons
  - Tools
  - Dockpanes
  - Embeddable control
  - ..
- Packaged within a single, compressed file with an .esriaddinX file extension
  - c:%Homepath%\Documents\ArcGIS\AddIns\ArcGISPro



What are the ArcGIS Pro Add-in core components?

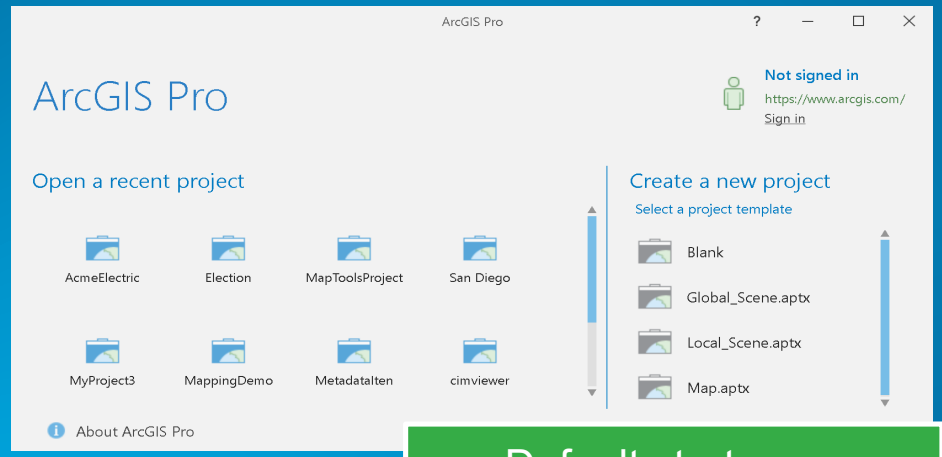
- **Declarative-based framework to define the UI elements**
  - Declarative framework is defined in a config.daml file
  - XML formatted, contains ArcGIS Pro framework elements (buttons, dockpane, galleries) and Add-in UI elements
- **The Module class**
  - Hub and central access point for each add-in
  - Similar to the Extension object used in the ArcObjects 10.x framework
  - Singletons instantiated automatically by the Framework


ArcGIS Pro - Election - Map
Analysis
View
Edit
Imagery
Share
Add-In
Map
Hudson Bay
Gulf of Alaska
Add-in Demo
Test
OK
Vancouver
Seattle
Chicago
Detroit
Denver
St Louis

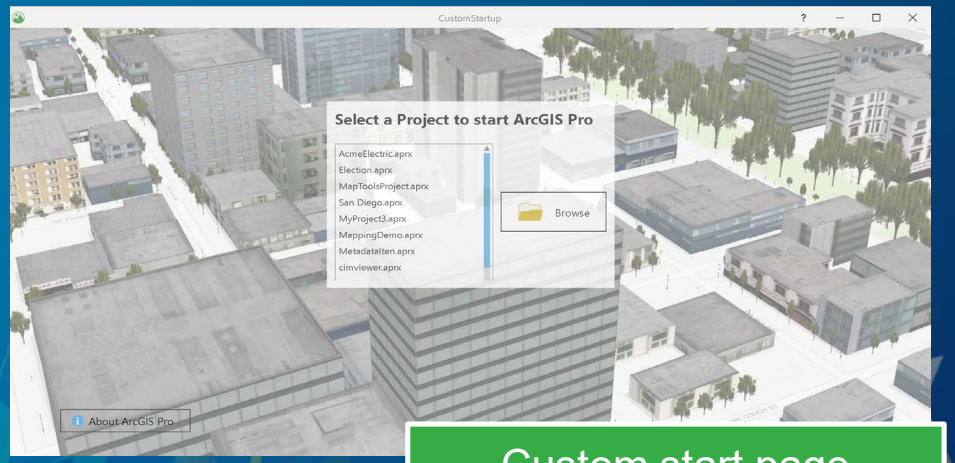
Demo: Add-in

What is an ArcGIS Pro Managed Configuration?

- **Includes all functionality of an add-in plus**
  - Change the application title and icon
  - Change the application splash, start page, and about page
  - Optionally customize and streamline the UI (the Pro Ribbon) for a specific workflow



Default start page



Custom start page

- **Packaged within a single, compressed file with a .ProConfigX file extension**
  - c:%Homepath%\Documents\ArcGIS\AddIns\ArcGISPro\Configurations

What is an ArcGIS Pro Managed Configuration?

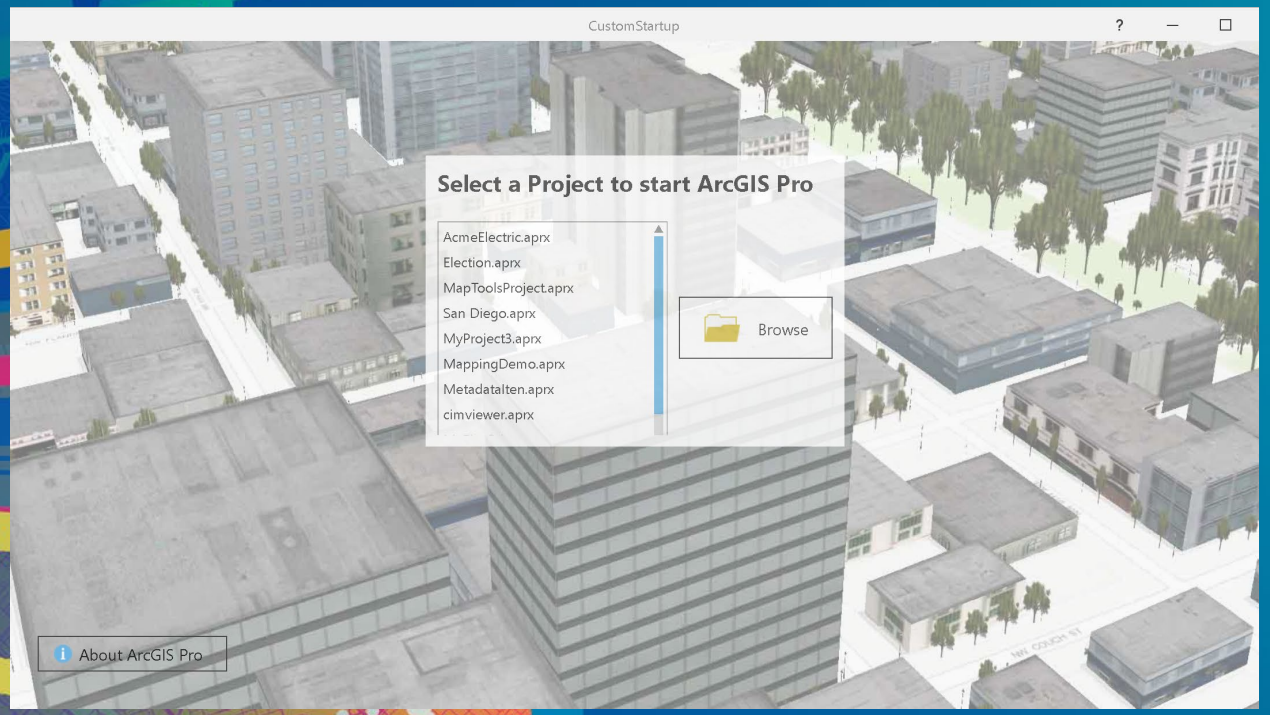
(Continued)

- Running an ArcGIS Pro Managed Configuration
  - Use the ArcGIS Pro "/config" command-line option

```
C:\ArcGIS\bin\ArcGISPro.exe /config:Acme
```

  - Only one configuration can run per instance of Pro
- ConfigurationManager class
  - Defined in DAML (generated automatically by the template)
  - Provides a set of methods by which a developer can *override* "that" aspect of Pro

```
public abstract class ConfigurationManager {
  protected internal virtual Window OnShowSplashScreen();
  protected internal virtual FrameworkElement OnShowStartPage();
  protected internal virtual FrameworkElement OnShowAboutPage();
  ...
```



Demo:

Configurations

Declarative Markup for Pro UI Elements: DAML

- ArcGIS Pro is using a “Declarative Markup” (DAML) language for all UI Elements
  - Buttons, Dockpane, Galleries, Property Sheets, Tools, …
- DAML is using XML Syntax, by default stored in a Config.daml file
- Using DAML you can add, modify, delete any User Interface element
- Pro is using DAML, look in the bin\Extensions folder

```
<ArcGIS defaultAssembly="WorkingWithDAML.dll" defaultNamespace="W
xmlns="http://schemas.esri.com/DADF/Registry"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.esri.com/DADF/Registry
file:///C:/Program%20Files/ArcGIS/Pro/bin/ArcGIS.Desktop.Framewor
  <AddInInfo id="{c6226a24-d69b-46c6-b5e7-9eee4ddad45d}" version=
…
</AddInInfo>
<modules>
    <insertModule id="WorkingWithDAML" className="Module1" autoLo
      <tabGroups>
        <!--The new Tab Group is created here-->
        <tabGroup caption="Example State Solution" id="working_wi
          <color A="255" R="238" G="170" B="90" />
          <borderColor A="0" R="251" G="226" B="195" />
        </tabGroup>
      </tabGroups>
```

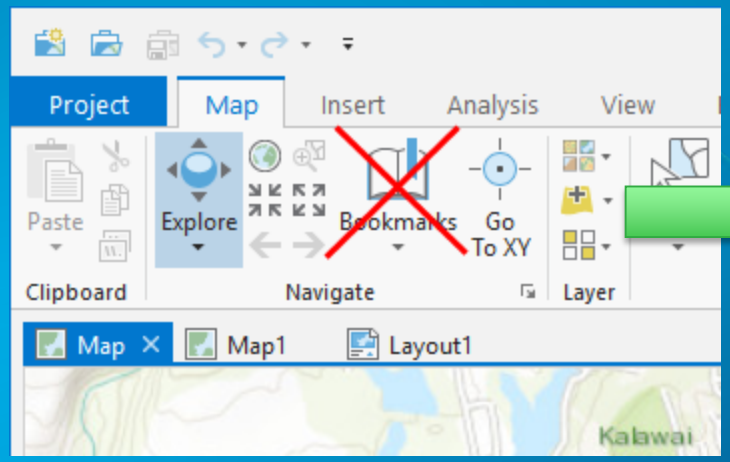
This PC › OSDisk (C:) › Program Files › ArcGIS › Pro › bin › Extensions

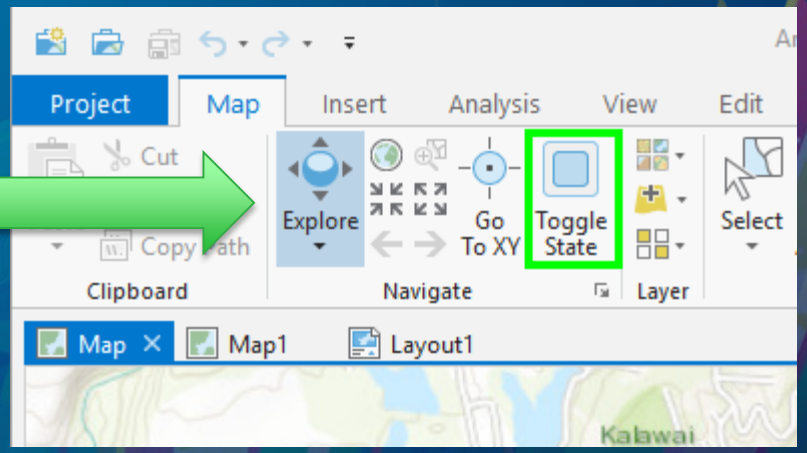
Name	Date modified	Type
Analyst3D	2/21/2019 10:17 A…	File folder
ArcGISSearch	2/21/2019 10:17 A…	File folder
Aviation	2/21/2019 10:17 A…	File folder
BusinessAnalyst	2/21/2019 10:17 A…	File folder
Catalog	2/21/2019 10:17 A…	File folder
Charts	2/21/2019 10:17 A…	File folder
Core	2/21/2019 10:17 A…	File folder
DataReviewer	2/21/2019 10:17 A…	File folder
DataSourcesRaster	2/21/2019 10:17 A…	File folder
DefenseMapping	2/21/2019 10:17 A…	File folder
DesktopExtensions	2/21/2019 10:17 A…	File folder
Editing	2/21/2019 10:17 A…	File folder
FullMotionVideo	2/21/2019 10:17 A…	File folder
GeoProcessing	2/21/2019 10:17 A…	File folder
GeostatisticalAnalysis	2/21/2019 10:17 A…	File folder

Declarative Markup for Pro UI Elements: DAML

- DAML is transactional
  - 3 distinct actions: Insert, Update, and Delete
  - The type of action is determined by the element name: updateModule, updateGroup, deleteButton

```
<updateModule refID="esri_mapping">
     <groups>
       <updateGroup refID="esri_mapping_navigateGroup">
         <deleteButton refID="esri_mapping_bookmarksNavigateGallery“ />
         <insertButton refID="working_with_DAML_ToggleStateButton" />
       </updateGroup>
     </groups>
</updateModule>
```





How to use DAML to change the UI of other Add-ins or existing Pro UI

- DAML is transactional and is processed in the order the Pro Extension or Add-in is loaded
- Transaction Example – the “esri_mapping_zoomFullButton” button
  - Mapping Extension is loaded and the ‘zoomFullButton’ is added

```
<group id="esri_mapping_navigateGroup" caption="Navigate" launcherButtonID=… >
     <tool refID="esri_mapping_exploreSplitButton" size="large" />
     <button refID="esri_mapping_zoomFullButton" size="small" />
     …
     <button refID="esri_mapping_gotoXYControl" size="large"/>
</group>
```

  - Custom Add-in is loaded and the ‘zoomFullButton’ is deleted

```
<updateGroup refID="esri_mapping_navigateGroup">
    <deleteButton refID="esri_mapping_zoomFullButton“ />
</updateGroup>
```

- In the DAML transactional model the ‘last’ transaction wins !

How to use DAML to change the UI of other Add-ins or existing Pro UI

- Ensuring the proper DAML processing order
- If your Add-in changes another Add-in or a Pro Extension
  - Use the Dependencies tag in your DAML
  - Add-in Example that changes the Mapping Extension has ‘dependency’ on ADMapping.daml

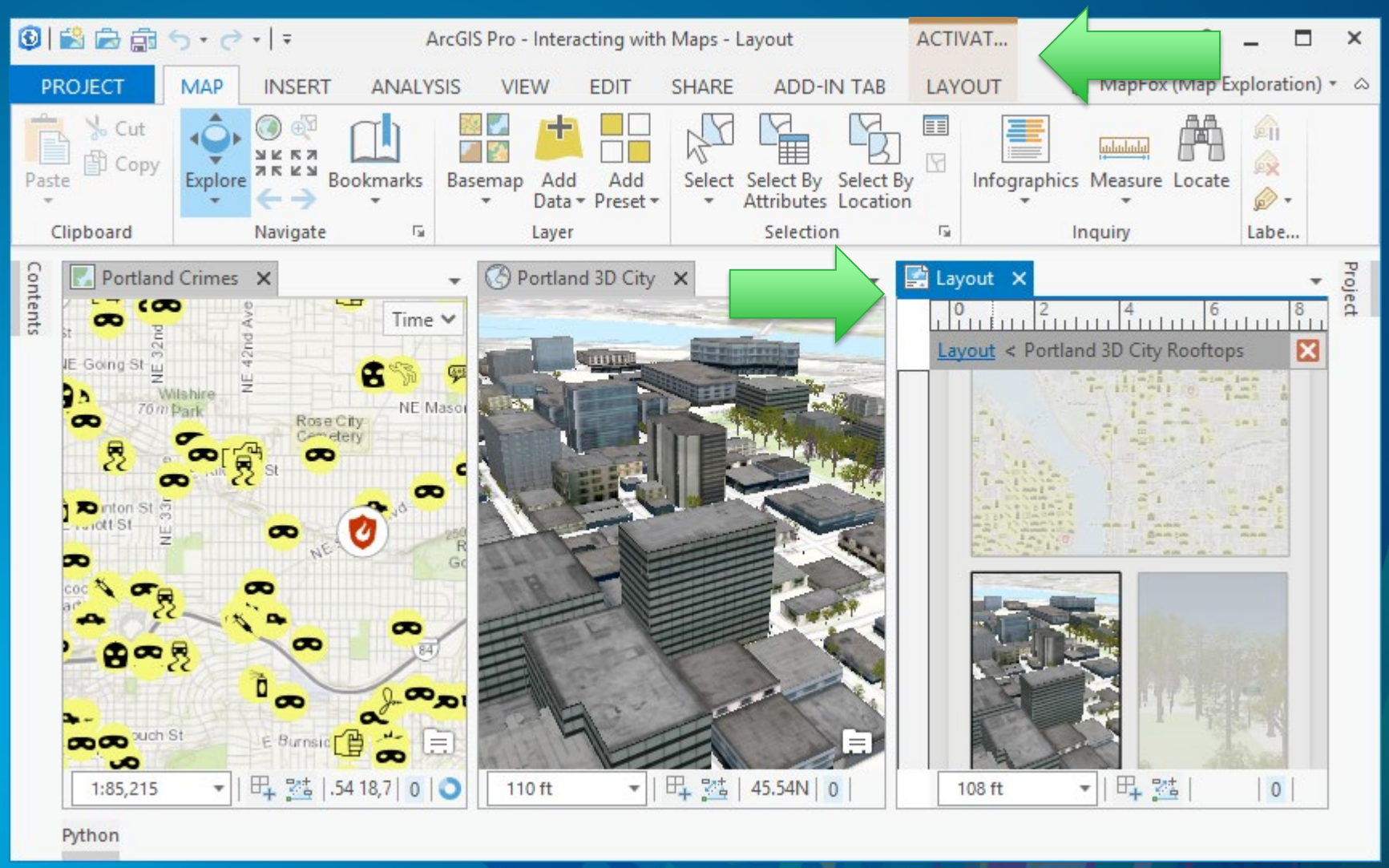
```
<dependencies>
   <dependency name="ADMapping.daml" />
</dependencies>
```

  - Add-in Example that changes another Add-in (using the “other” Add-in’s AddInInfo tag’s id attribute as identifier)

```
<dependencies>
   <dependency name="{c1a60c8f-2f6f-4198-a5d6-ea964ebf678c}" />
</dependencies>
```

States & Conditions

- ArcGIS Pro UI is contextual – Implemented through “States & Conditions”



States & Conditions

- States & Conditions are used to simplify coding
  - No event wiring as with more traditional models.
  - Example: A button is enabled only when a “sample_state_condition” is met.
- States
  - are named values describing a particular aspect of the application’s overall status.
  - Activate or deactivate States in code
- Conditions
  - Declared in DAML
  - Expressions composed of one or more states
  - AND, OR, NOT
  - Used for triggering the activation and deactivation of framework UI elements.

States and Conditions (continued)

```
<conditions>
   <insertCondition id="example_state_condition" caption="Custom Condition">
      <state id="example_state" />
   </insertCondition>
</conditions>
```

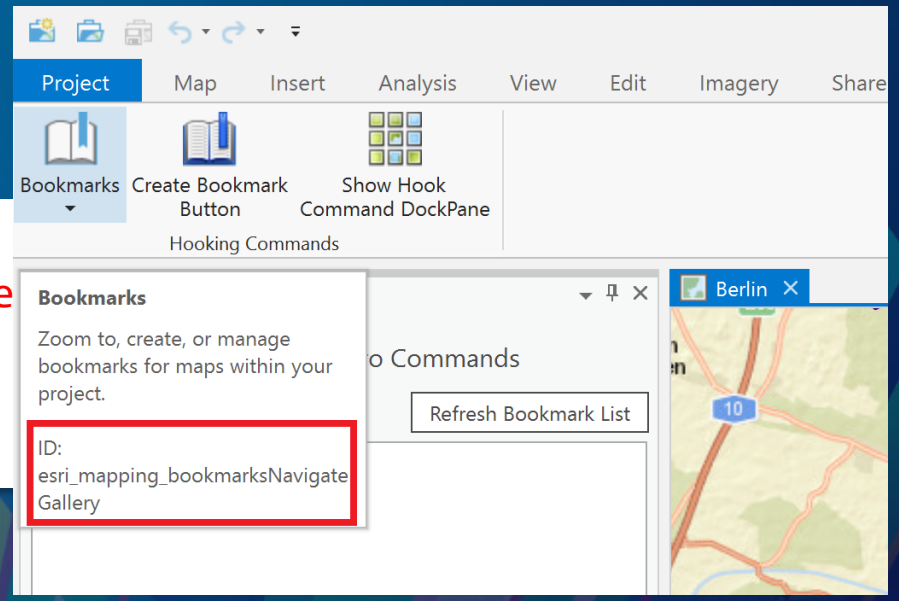
```
if(FrameworkApplication.State.Contains("example_state")
      FrameworkApplication.State.Deactivate("example_state");
else
      FrameworkApplication.State.Activate("example_state");
```

```
<!-- associate our condition with the enabled state of the button -->
<button id="esri_sdk_RespondToAppStateBtn" caption="Respond to state"
        condition="example_state_condition">
</button>
```

How to use existing Pro UI elements in Your Add-in

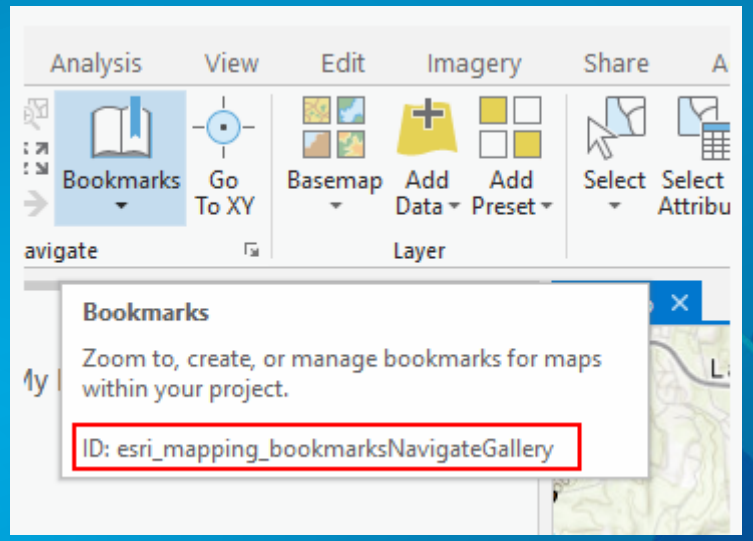
- Use any existing ArcGIS Pro framework elements on your Add-in tab including Buttons, Tools, Galleries
- Find the Unique Element Id of any existing ArcGIS Pro Control and use the Id in your config.daml markup to add, delete, modify that Element
- Example: Add Pro's Navigate Bookmarks button to my Add-in toolbar
  - button tag references existing Element ID: esri_mapping_bookmarksNavigateGallery

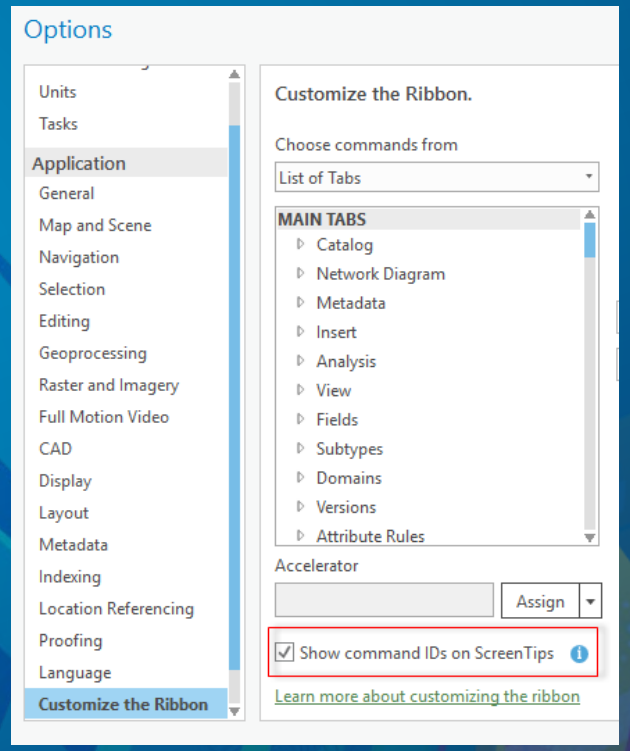
```
<group id="HookingCommands_Group1" caption="Hooking Commands" ke
   <button refID="esri_mapping_bookmarksNavigateGallery" />
</group>
```

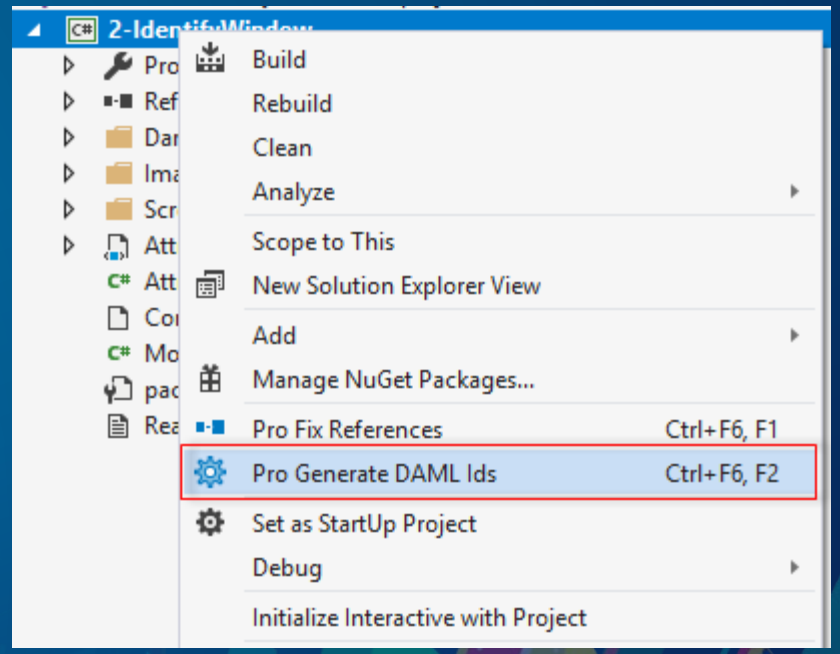


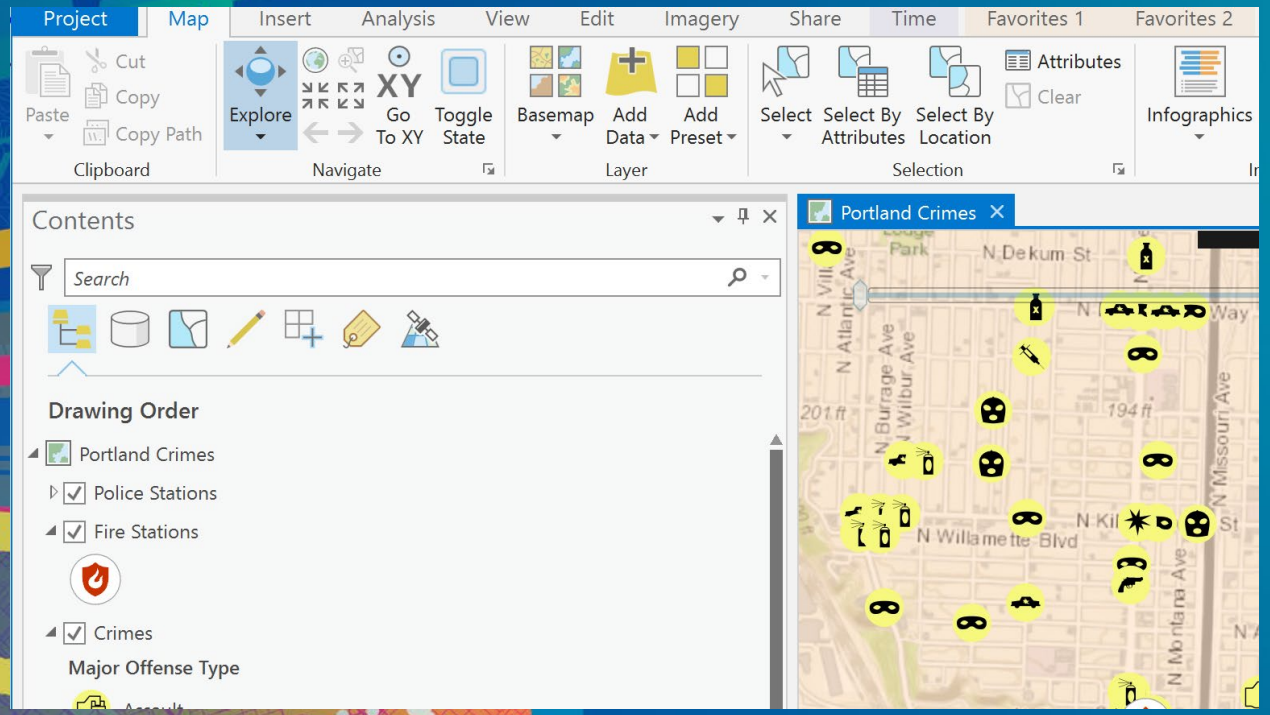
How to find Exiting ArcGIS Pro Element Ids

- **SDK Help: ArcGIS Pro DAML ID Reference**
  https://github.com/Esri/arcgis-pro-sdk/wiki/ArcGIS-Pro-DAML-ID-Reference
- **Pro Generate DAML Ids tool in Visual Studio**
  - **Allows Intellisense to find Ids:**
    i.e. DAML.Button.esri_mapping_addDataButton
- **ArcGIS Pro 'Customize the Ribbon': Show IDs option**







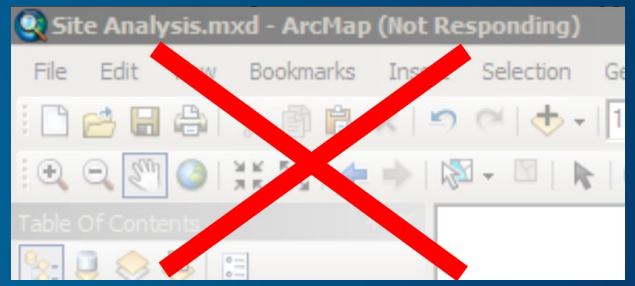


Demo: Add-in using ArcGIS Pro Buttons, States & Conditions

WorkingWithDAML project

Asynchronous Programming

- ArcGIS Pro is a multi-threaded 64 bit application
- Main motivation to use Asynchronous Programming is to keep the User Interface responsive !



- There are two important asynchronous programming patterns for the SDK:
  - Use of Async / Await functionality in C# or VB .Net
  - Using the ArcGIS Pro Framework's QueuedTask class

ArcGIS Pro Internal Threading Model

- **ArcGIS Pro multi-threading:**
  - Incorporates the latest asynchronous language features from Microsoft

- **ArcGIS Pro SDK developers only need to worry about two threads:**
  - The GUI thread (Graphical User Interface thread)
  - A single specialized worker thread called the Main CIM Thread, MCT

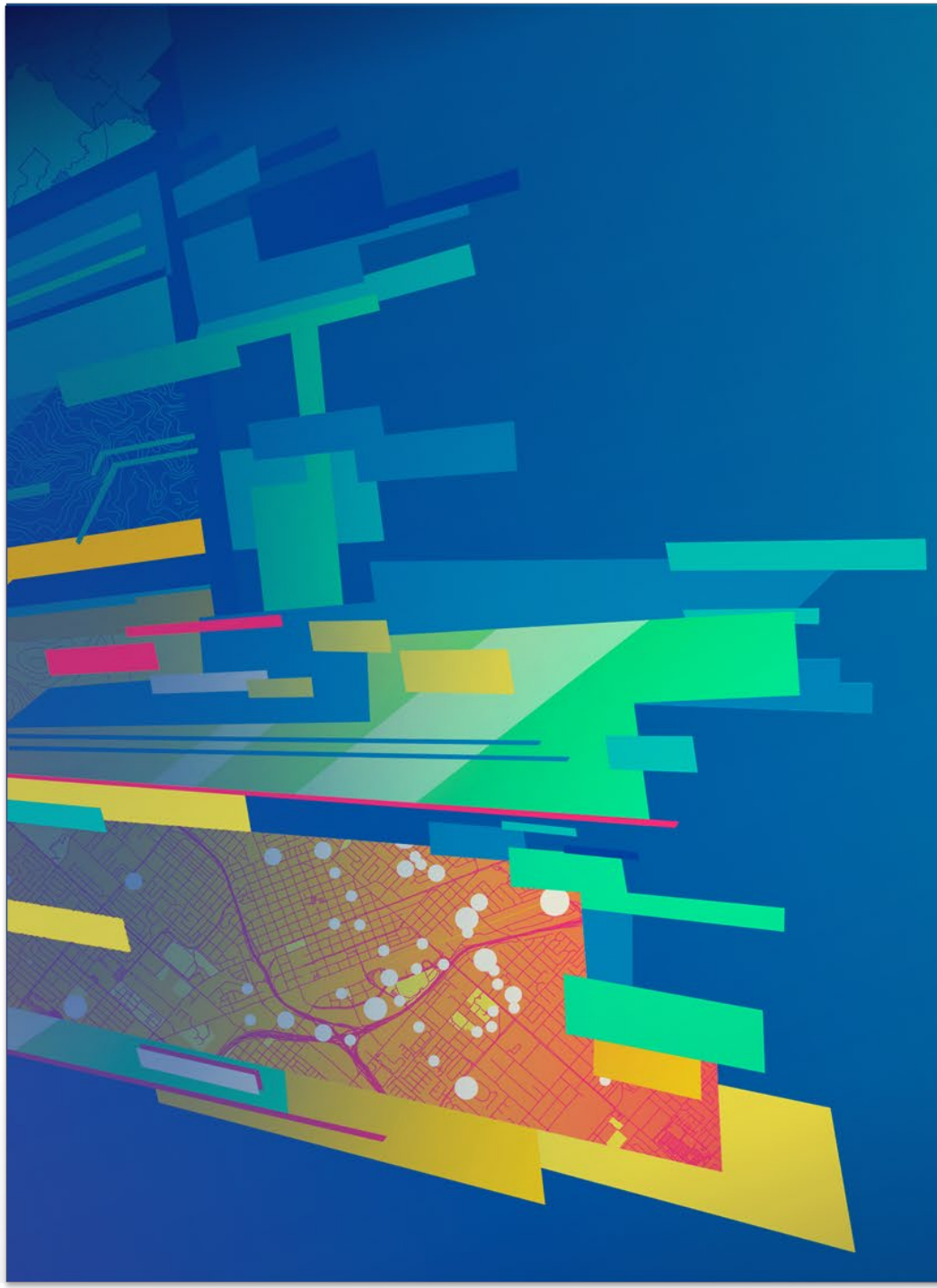
Asynchronous Programming

- There are 2 Categories of Methods in the ArcGIS Pro API
  - Coarse-grained asynchronous methods:
    - Can be called on any thread
    - Once called these methods return to the caller immediately but execute asynchronously in the background
  - Fine-grained synchronous methods:
    - Must be called using the QueuedTask class
    - These methods are ‘blocking’ and return to the caller after the method is finished doing its work

Coarse-Grained, Asynchronous Methods

- Can be called from any thread. Typically invoked from the UI thread
- Coarse-grained methods execute in the background on Pro internal threads
- Coarse-grained methods use the ‘Async’ naming convention:
  - method names end with ‘Async’
- Caller uses the async / await semantic

```
//Execute a Geoprocessing Tool
await Geoprocessing.ExecuteToolAsync("SelectLayerByAttribute_management",
     new string[] {"parcels","NEW_SELECTION", "description = 'VACANT LAND'"});
await MapView.Active.ZoomToSelectedAsync(new TimeSpan(0, 0, 3));
```



Demo: Coarse Grained Methods

PlenaryPopup Data

Fine-Grained, Synchronous Methods

Fine-grained Synchronous Methods Must be called using the QueuedTask class

- There is a <u>much greater number</u> of fine grained methods and classes
- No async / await. Runs on the MCT (Main CIM Thread managed by ArcGIS Pro)
- Designed so you can write your business logic as a ‘background’ task

```
await ArcGIS.Desktop.Framework.Threading.Tasks.QueuedTask.Run(() =>
{
   var layers = MapView.Active.Map.FindLayers("Parcels")
                 .OfType<FeatureLayer>().ToList();
   var parcels = layers[0] as FeatureLayer;
   QueryFilter qf = new QueryFilter()
   {
       WhereClause = "description = 'VACANT LAND'",
       SubFields = "*"
   };
   parcels.Select(qf, SelectionCombinationMethod.New);
});
```

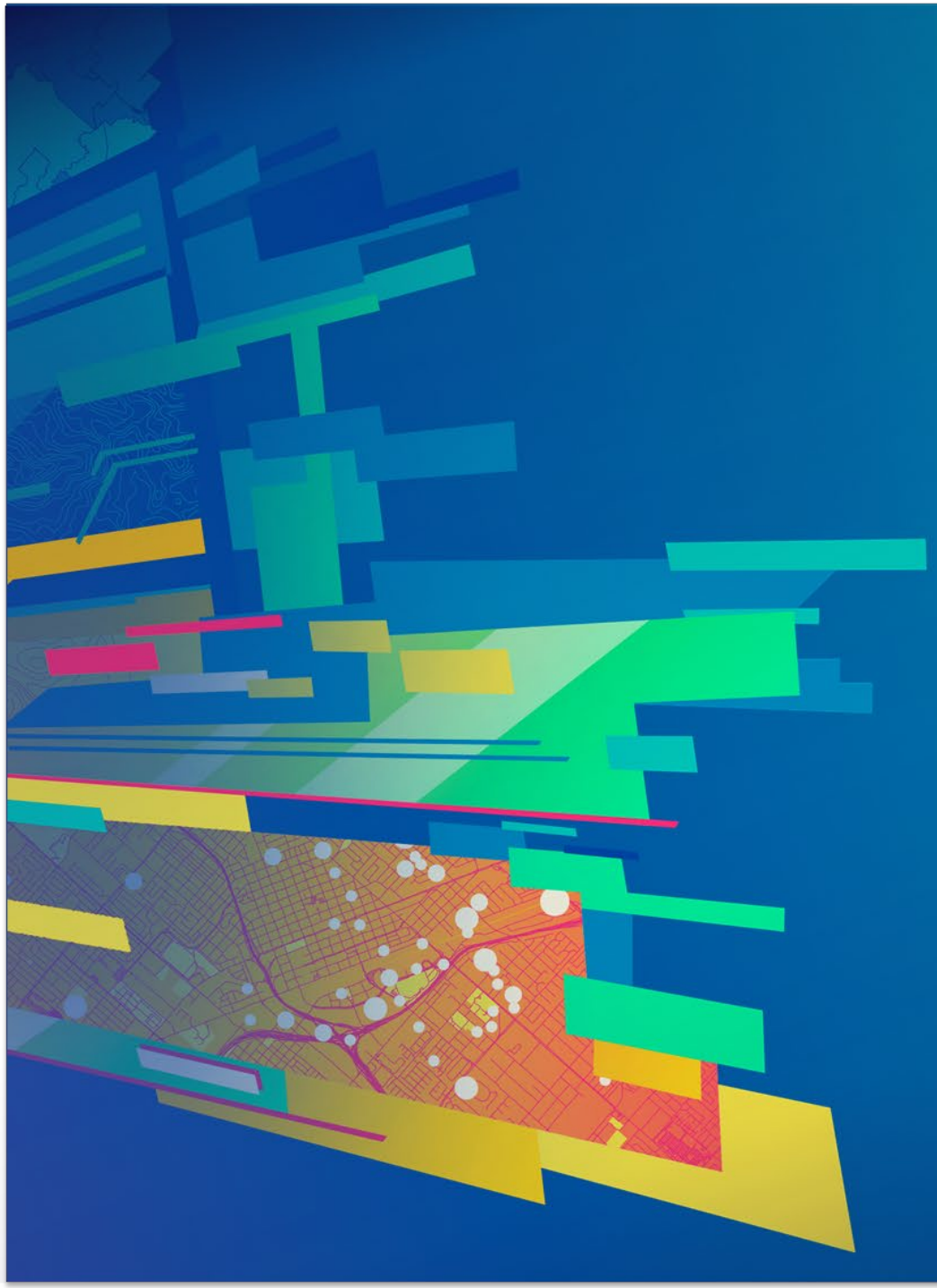
QueuedTask Class

- QueuedTask uses the Pro framework's custom Task scheduler
- Used to run 'blocking' ArcGIS Pro SDK methods on a background worker thread
- Synchronous API methods are listed in the API Reference guide using the following text in the description:

  "This method must be called on the MCT. Use QueuedTask.Run"

- Example of synchronous methods in Pro:
  - GetSpatialReference, QueryExtent, all Geometry operations
- Usage:

```
Task t = QueuedTask.Run(() =>
{
    // Call synchronous SDK methods here
});
```



Demo: Fine-Grained Methods

- PlenaryPopup Data

ArcGIS Pro SDK for .NET – Technical Sessions

Date	Time	Session	Location
Tue, Mar 05	4:00 pm - 5:00 pm	Beginning Pro Customization with focus on DAML and Customization Patterns	Mojave Learning Center
	5:30 pm - 6:30 pm	Beginning Editing with Focus on EditOperation	Primrose A
Wed, Mar 06	10:30 am - 11:30 am	Understanding the CIM, a Guide for Developers	Mesquite C
	2:30 pm - 3:30 pm	Intermediate Pro Customization with focus on Mapping and Layout APIs	Mesquite C
	5:30 pm - 6:30 pm	Intermediate Editing with Focus on UI Customizations	Mesquite G-H
Thu, Mar 07	9:00 am – 10:00 am	Understanding Feature Services, a Guide for Developers	Primrose C-D
	10:30 am - 11:30 am	An Overview of the Utility Network Management API	San Jacinto
	1:00 pm – 2:00 pm	An Overview of the Geodatabase API	Mesquite G-H
	5:30 pm - 6:30 pm	Advanced Pro Customization with focus on Categories and Custom Settings	Smoketree A-E
Fri, Mar 08	8:30 am – 9:30 am	Advanced Editing with Focus on Edit Operations, Transaction Types, and Events	Mesquite C
	10:00 am - 11:00 am	Demonstrating Pro Extensibility with Add-Ins	Mesquite B

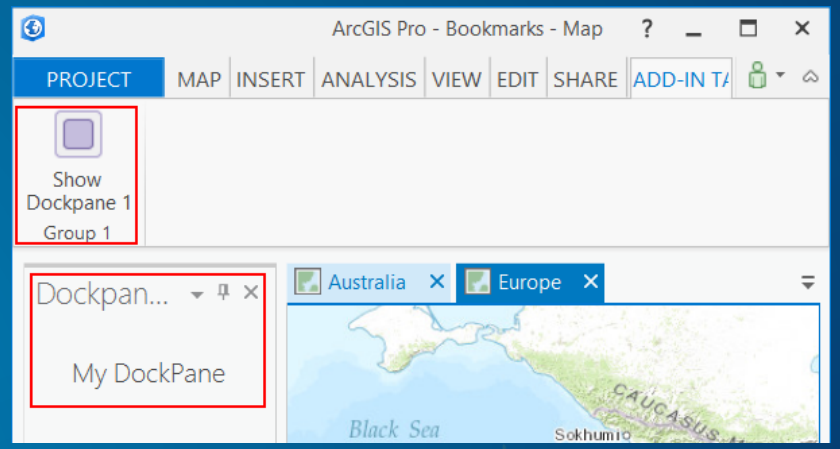
esri | THE SCIENCE OF WHERE

Session Overview

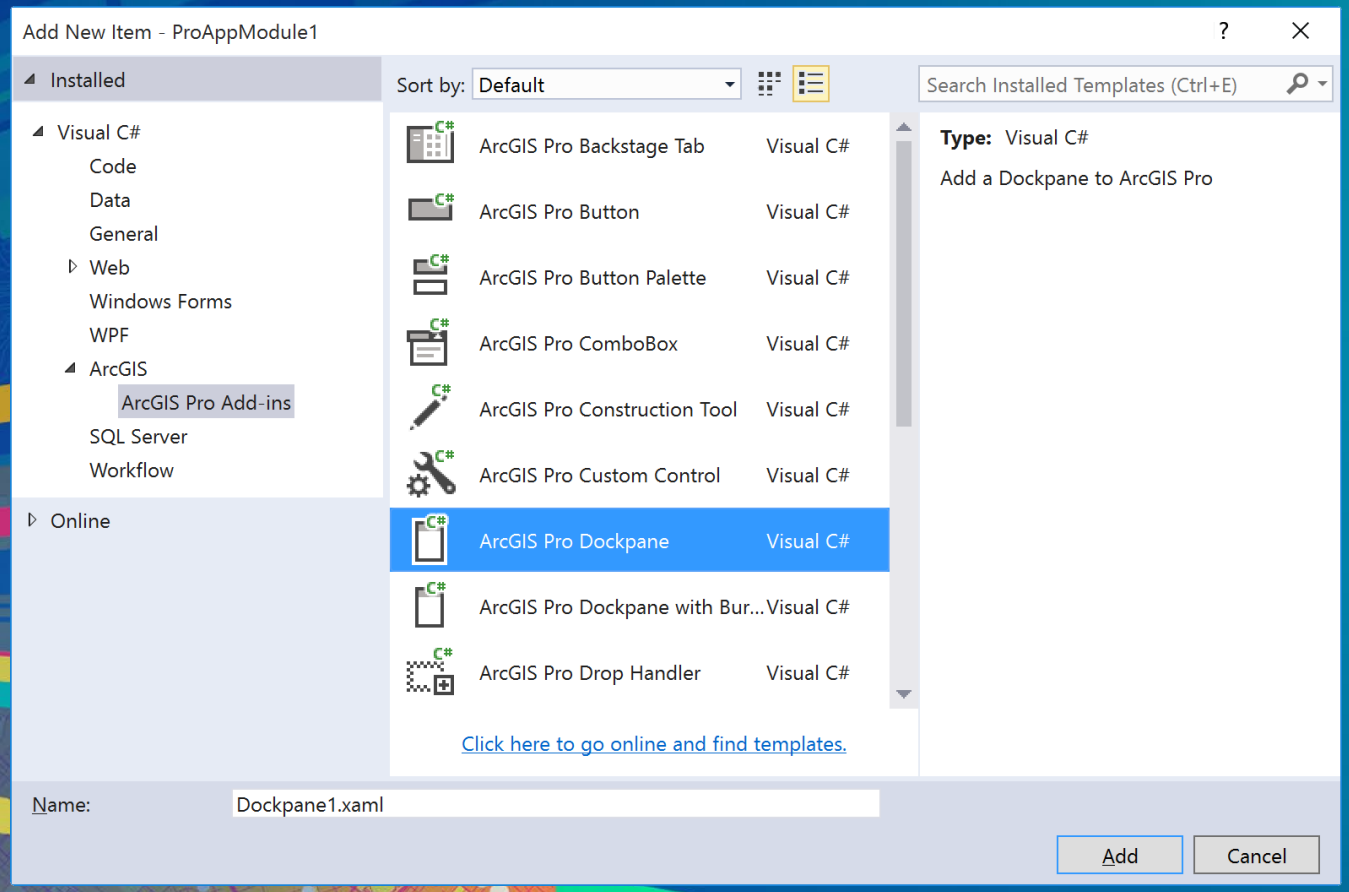
- The SDK is using Microsoft’s WPF and XAML in Add-ins
  - Dockpane example

WPF and XAML in Add-ins

- Many Framework Elements in ArcGIS Pro use Microsoft's WPF:
  - Dockpane
  - Pane
  - Custom Control
  - Embeddable Control
  - Property Page
  - …



- Implemented using the MVVM (model-view-viewmodel) Programming Pattern:
  - ViewModel: declared in DAML and implemented in code
  - View: referenced in DAML and implemented as WPF UserControl (XAML)
  - Model is optional



Demo: New Dockpane using MVVM Dockpane template

- PlenaryPopup Data