

Operations Dashboard for ArcGIS: Customizing and Extending

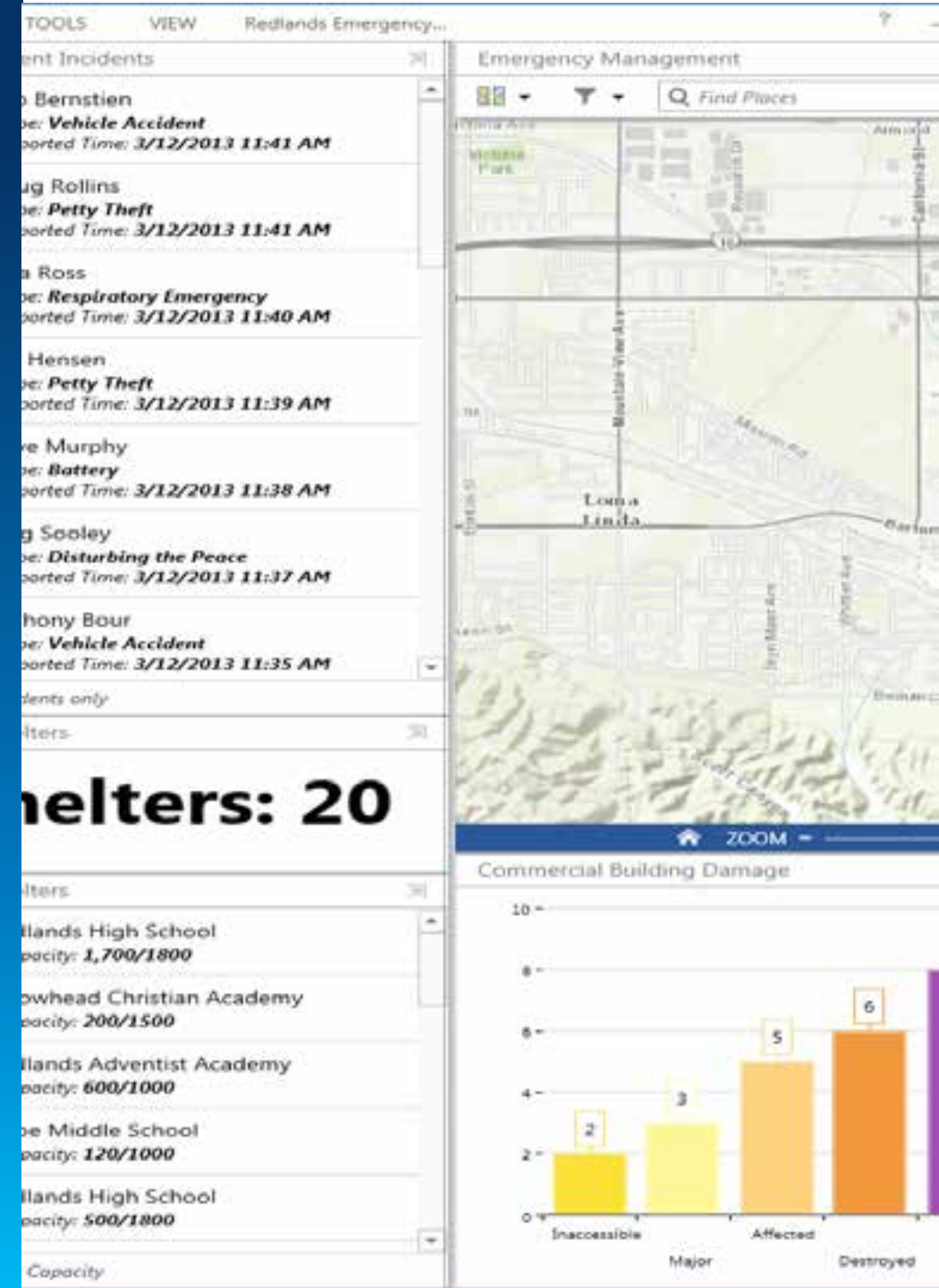
Sam Berg

sberg@esri.com

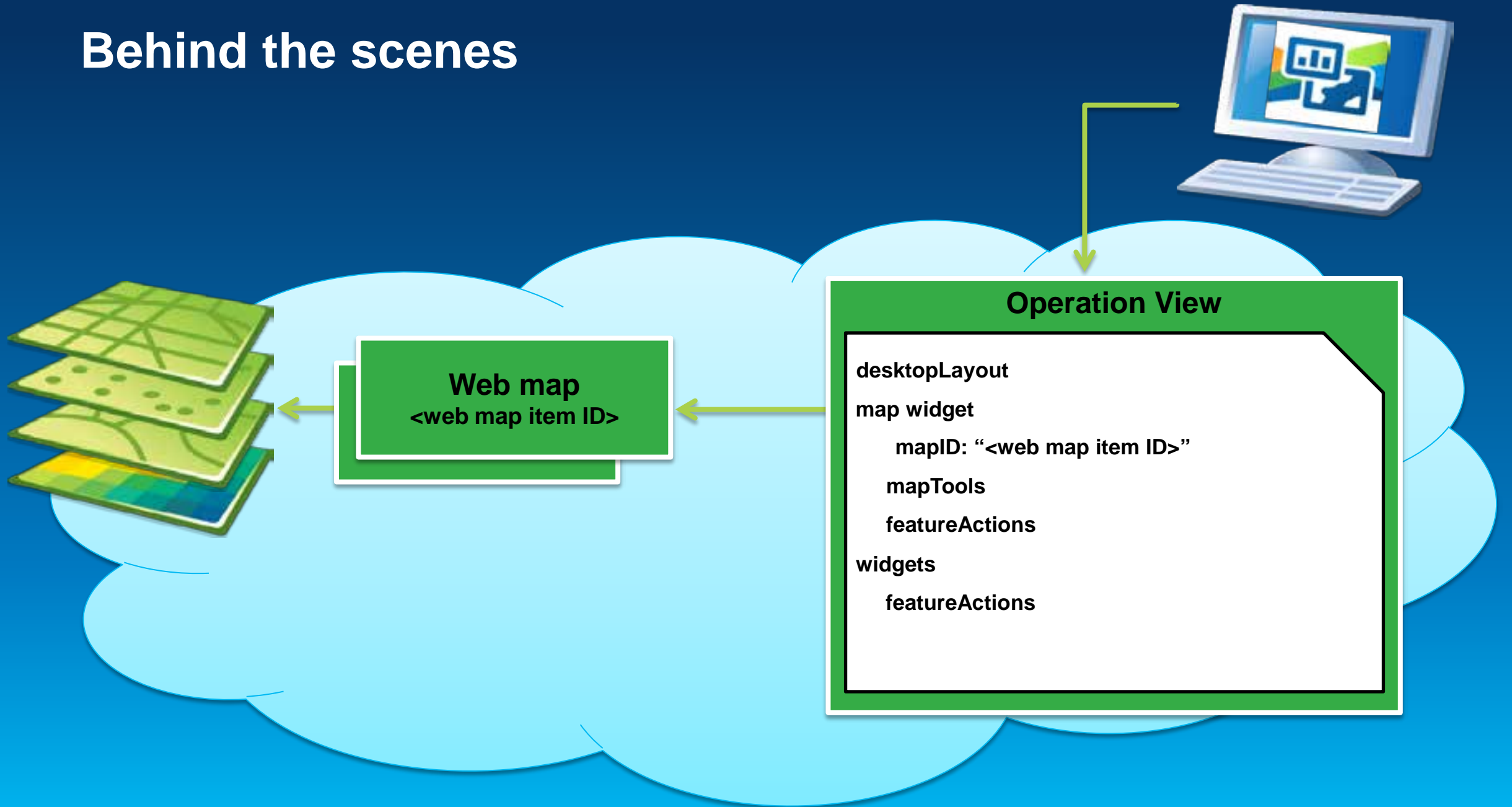
Agenda

- Operations Dashboard for ArcGIS Overview
- Add-Ins Overview
- Building a Widget
- Building a MapTool
- Building a FeatureAction
- Deploying an Add-In
- Updating an Add-In

Operations Dashboard for ArcGIS Overview



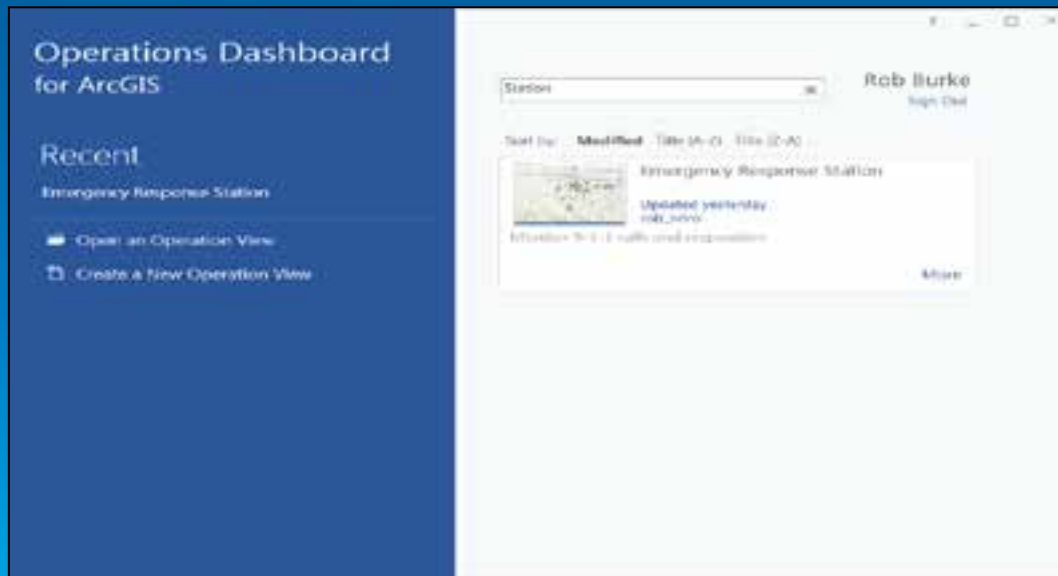
Behind the scenes



Interacting with operation views

- Users interact
- Publishers customize and configure
- Developers create new components

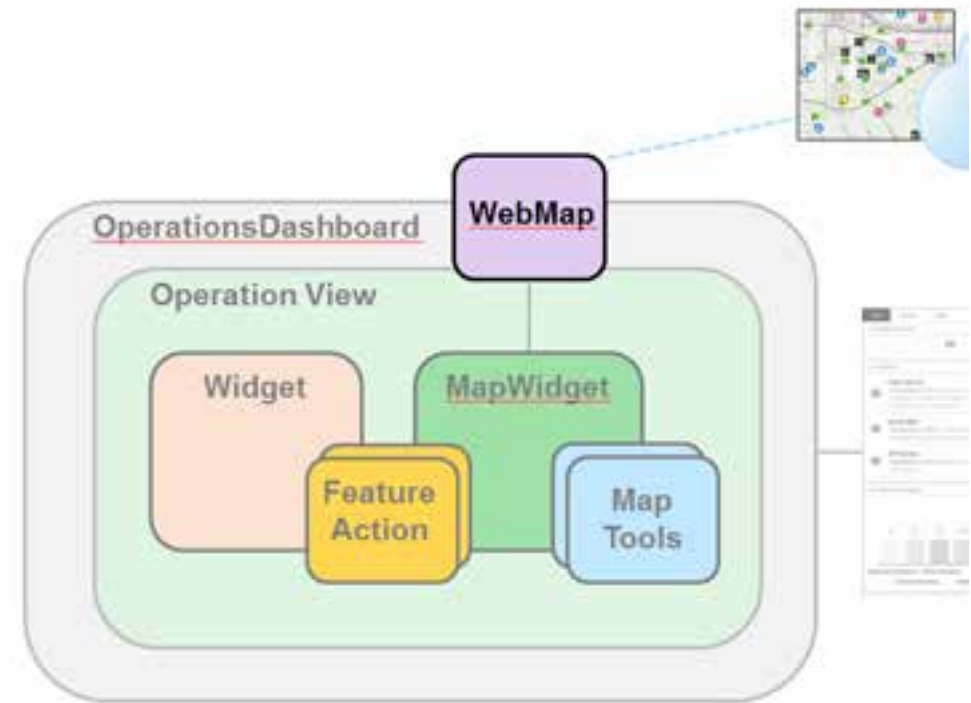
Operations Dashboard Application



Operation View



Operations Dashboard Add-Ins



Basics of add-ins

- Created with ArcGIS Runtime SDK for WPF
- Package together in a zip file
 - *.opdashboardAddin
- Share through ArcGIS Online or Portal for ArcGIS
 - Updates automatically

ArcGIS Runtime SDK for WPF

Communities

Home Concepts Sar

Extend and customize Operations Dashboard!

The ArcGIS Runtime SDK for WPF brings the extensibility you need to add widgets and other customizations to your Operations Dashboard.

[Read more](#)



Quick Links

Download and Install

- Download
- System requirements
- Installation
- Getting started with W

Help

- Concepts
- API reference
- Forum | Developer Bloc

Other links

- Ideas
- Training

Dashboard objects

ArcGIS



Dashboard objects

OperationsDashboard

ArcGIS



Dashboard objects

ArcGIS



OperationsDashboard

Operation View



Dashboard objects

ArcGIS



OperationsDashboard

Operation View

MapWidget



Dashboard objects

ArcGIS



OperationsDashboard

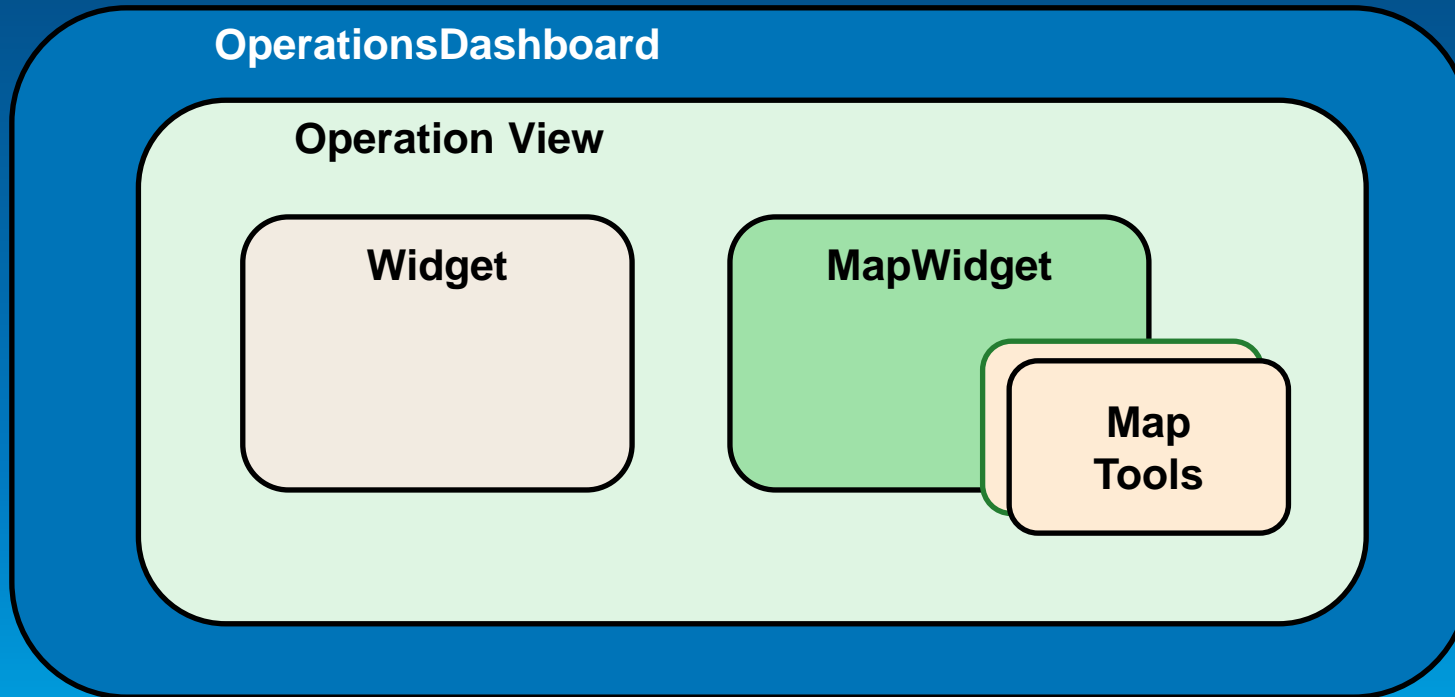
Operation View

Widget

MapWidget

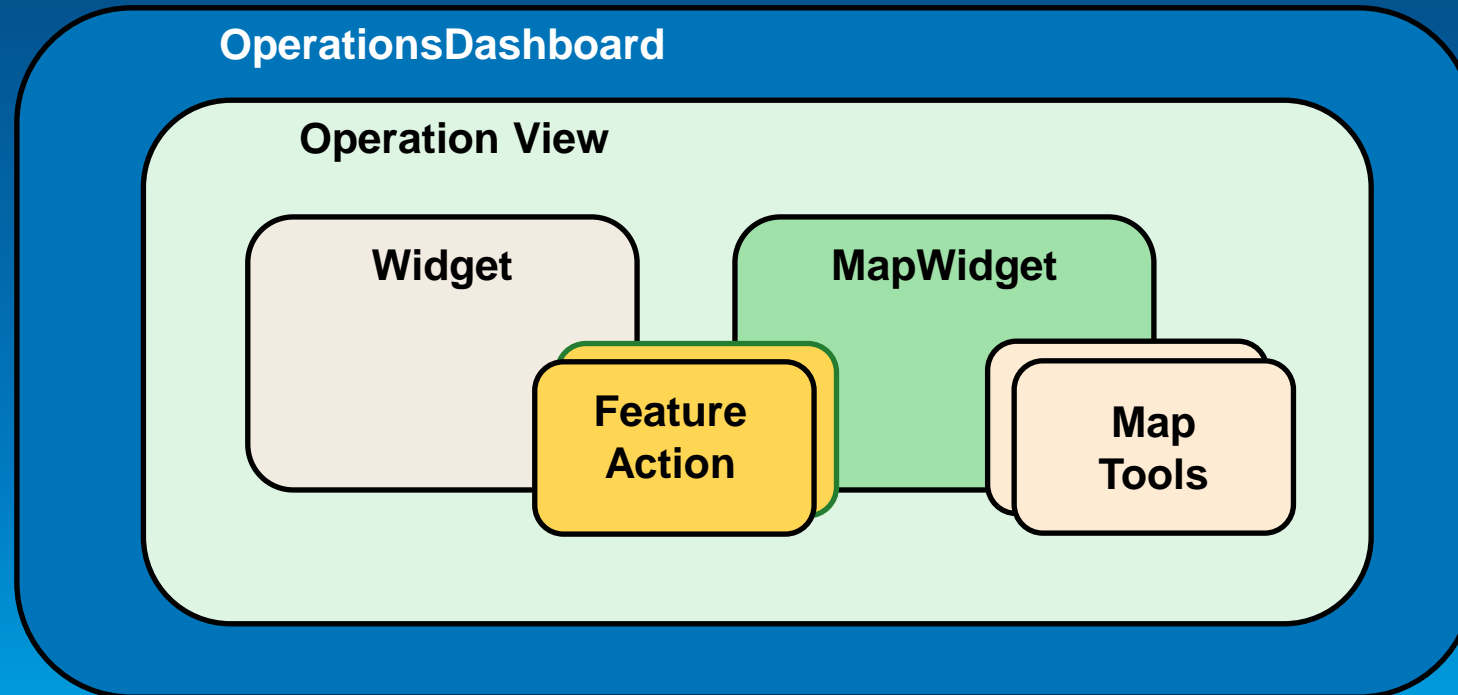


Dashboard objects



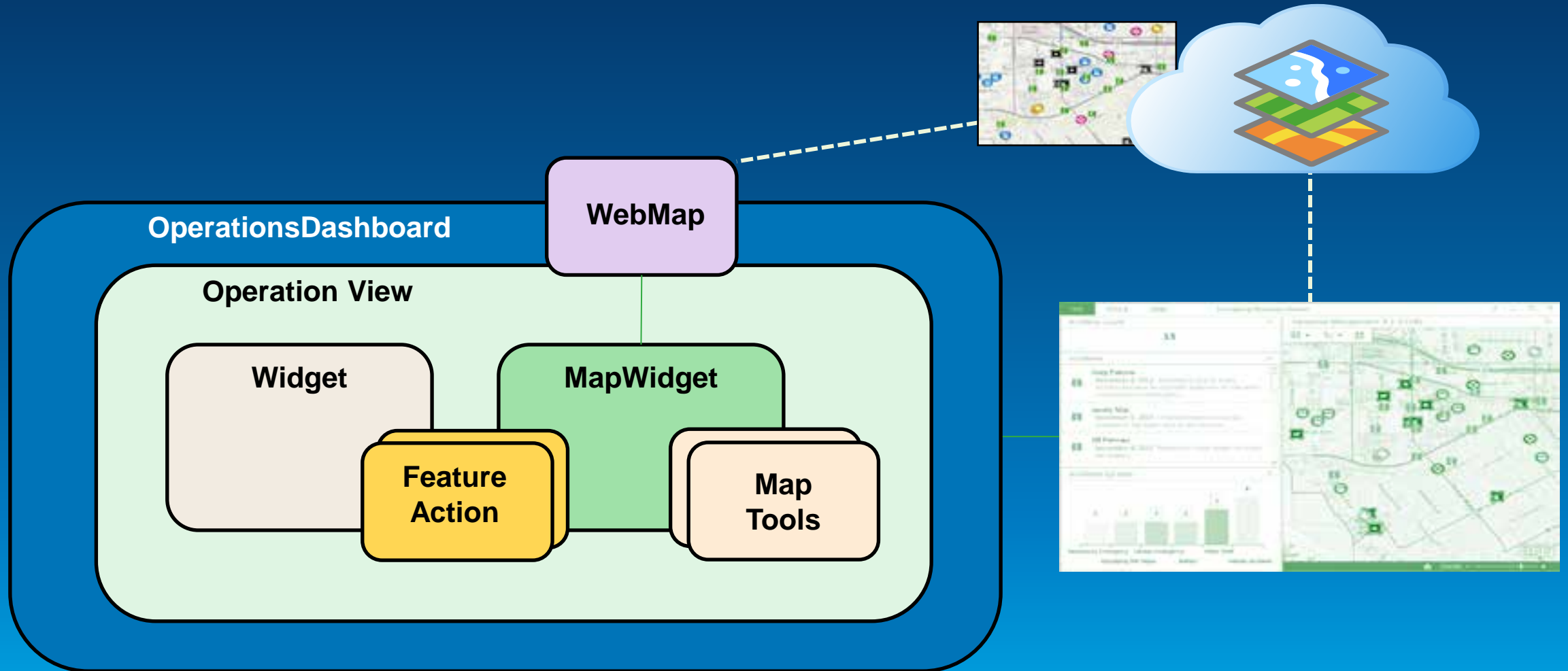
Dashboard objects

ArcGIS



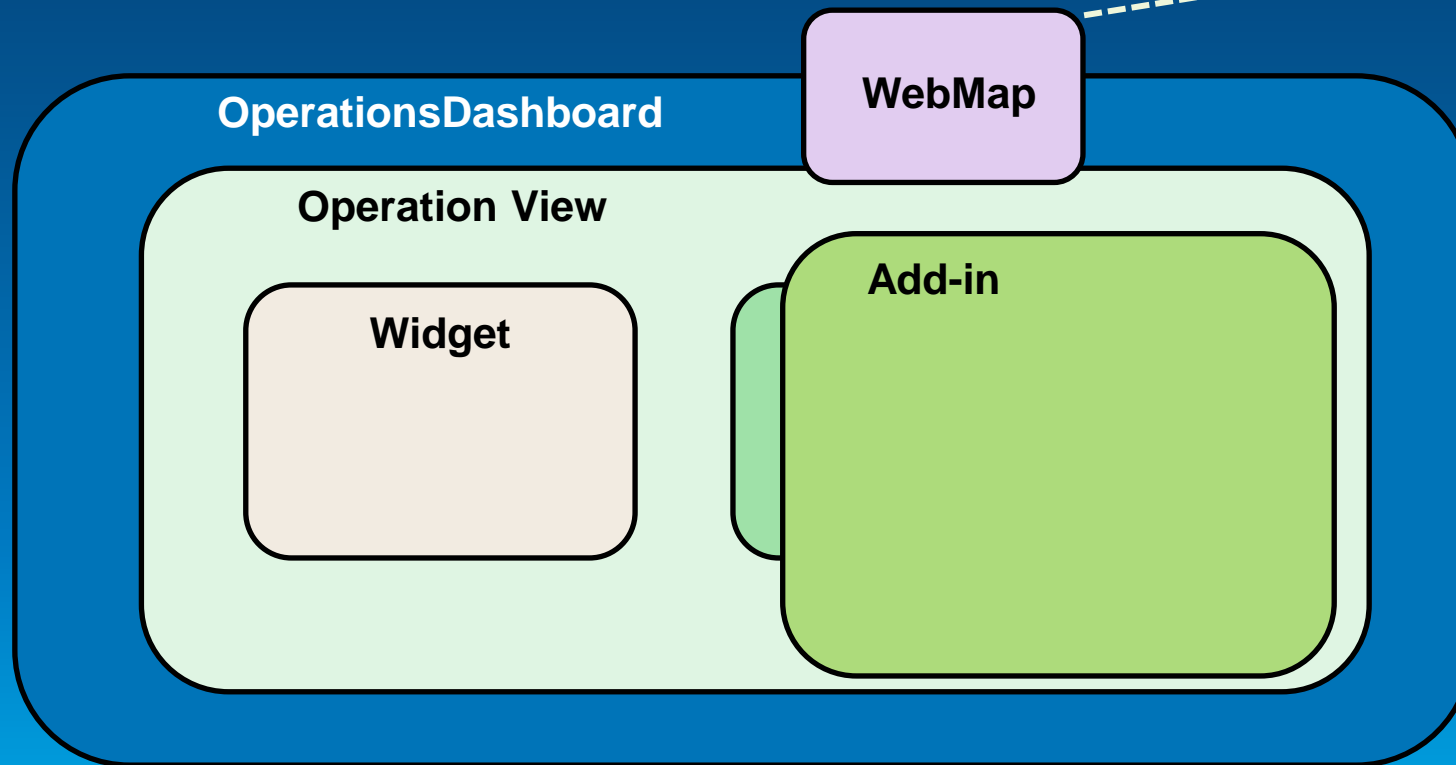
Dashboard objects

ArcGIS



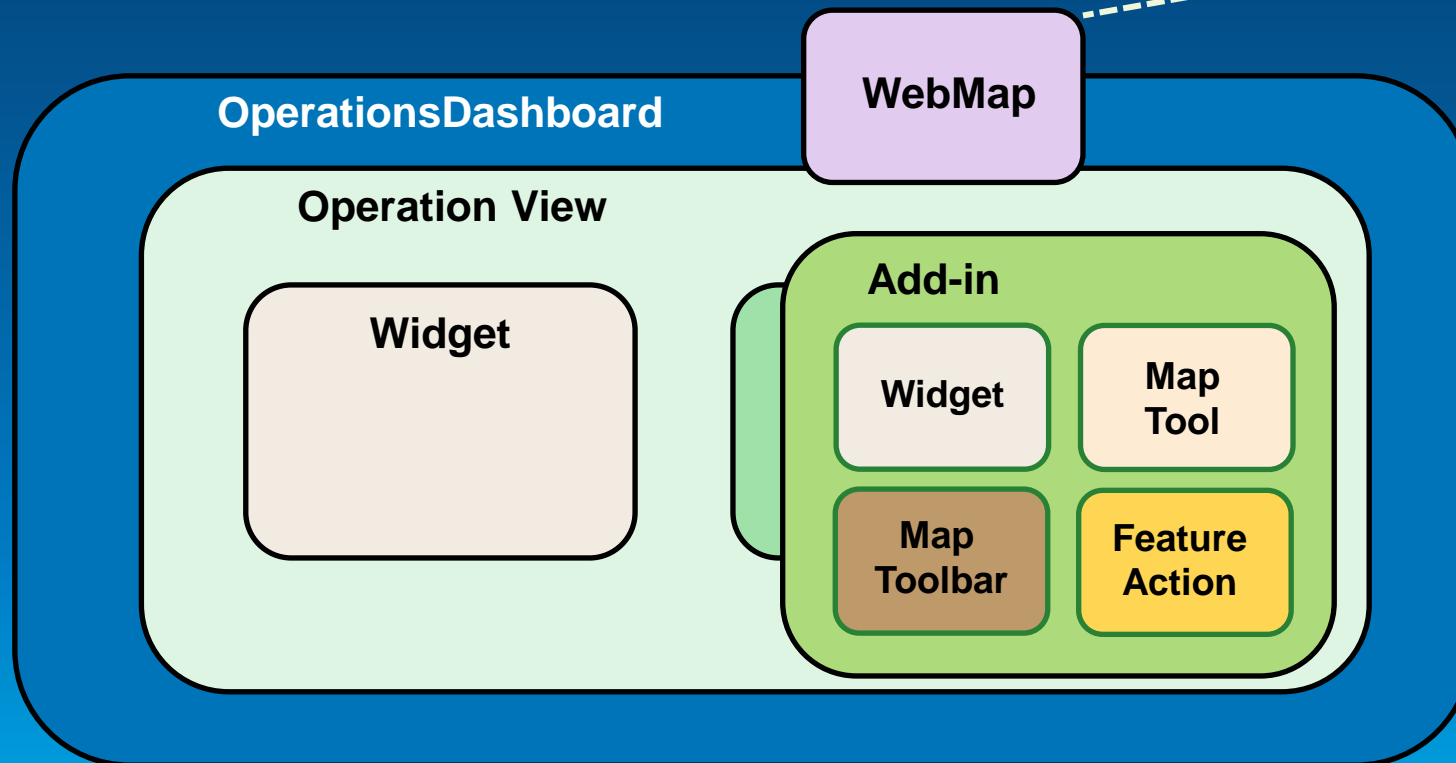
Dashboard objects

ArcGIS

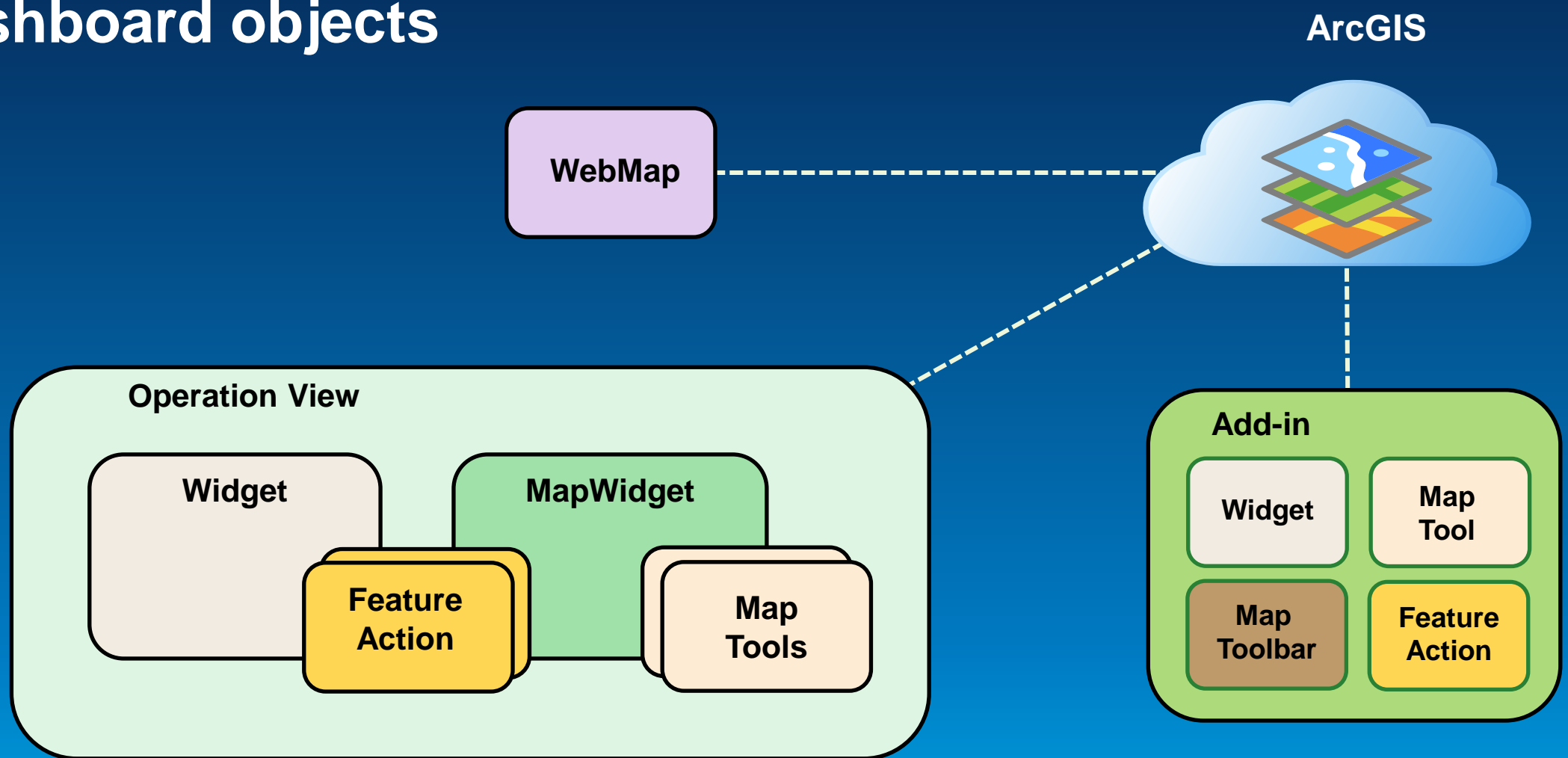


Dashboard objects

ArcGIS

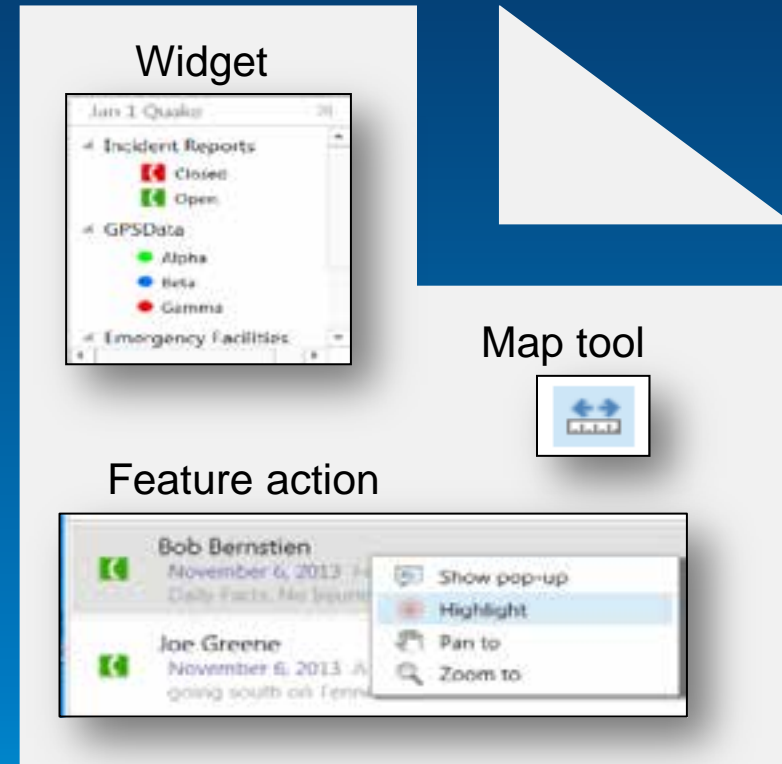


Dashboard objects



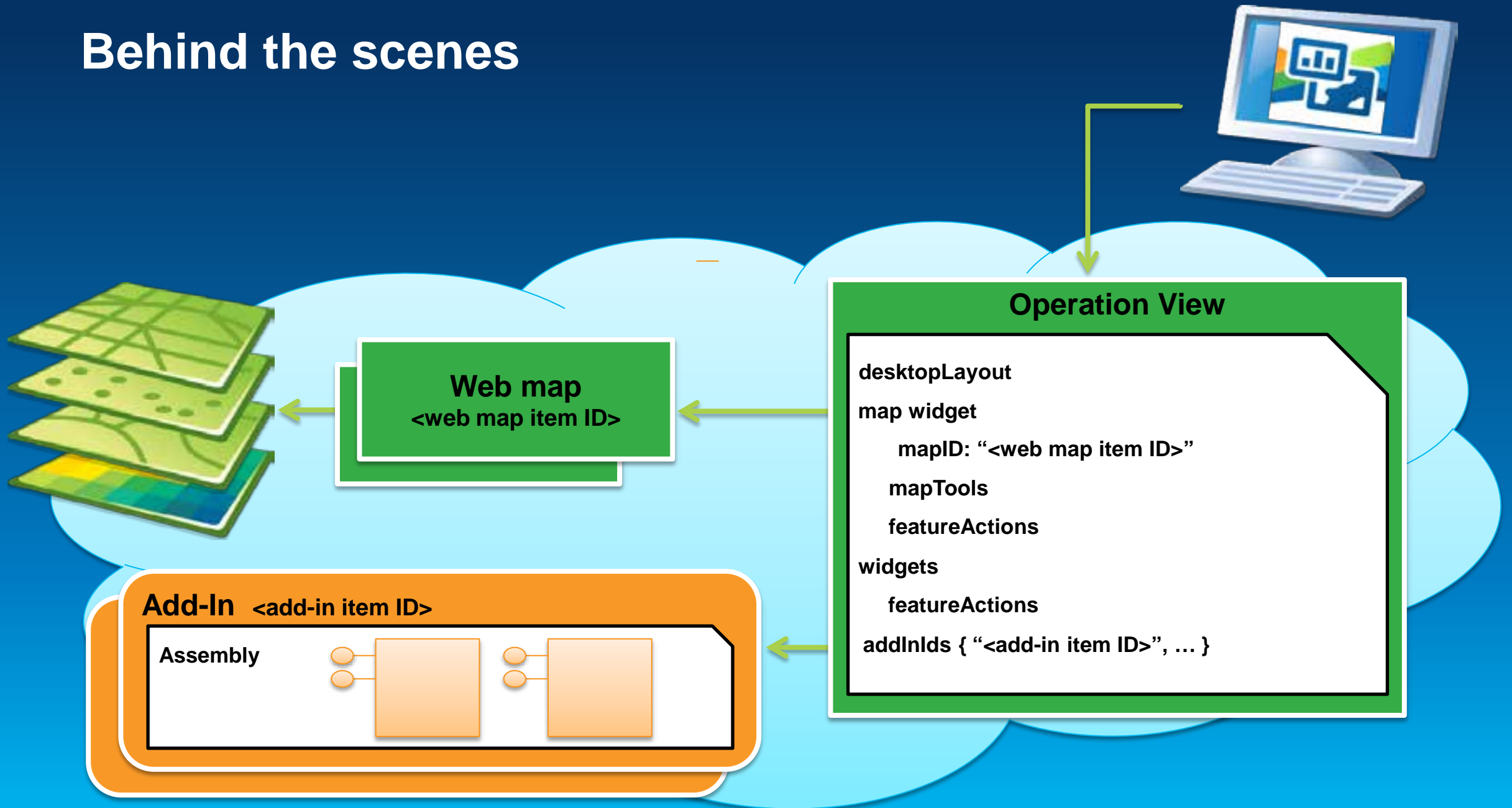
Add-ins contain your components

- Stored as a zip file
- Shared with organization
 - ArcGIS Online
 - Portal for ArcGIS
- Easy to update



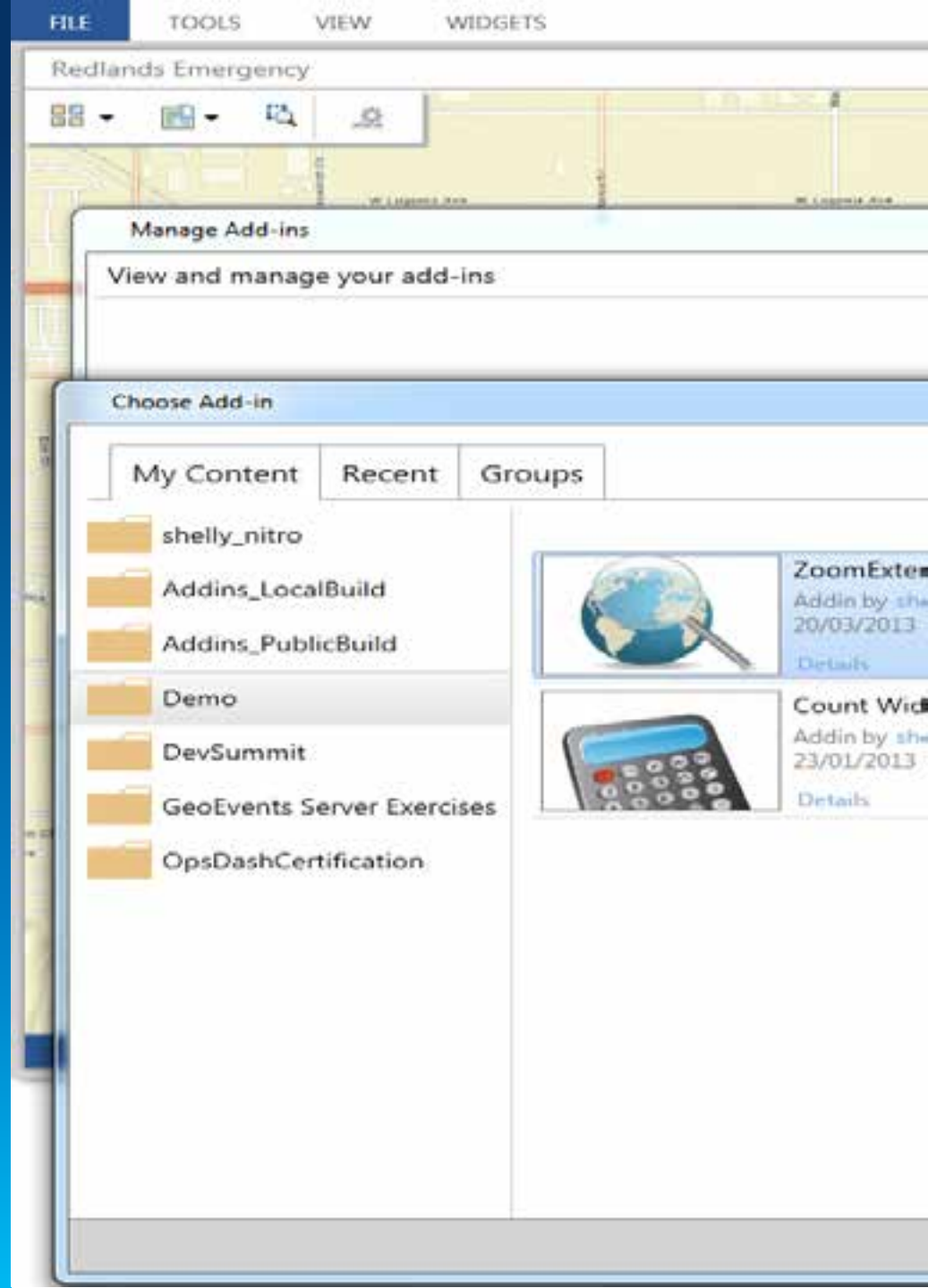
MyOps.opdashboardAddin

Behind the scenes



Demo

Find and use an add-in



Developing add-ins

The screenshot displays the Solution Explorer in Visual Studio for a project named 'OperationsDashboardAddIn1'. The project is a C# solution containing several folders and files. The 'Widget1' class is expanded to show its properties and methods.

Solution Explorer Structure:

- Solution 'OperationsDashboardAddIn1' (1 project)
 - OperationsDashboardAddIn1
 - Properties
 - References
 - Config
 - Widget1Dialog.xaml
 - Widget1Dialog.xaml.cs
 - Widget1Dialog
 - Images
 - Widget32.png
 - Widget1.xaml
 - Widget1.xaml.cs
 - Widget1
 - DataSourceBox : TextBox
 - FieldBox : TextBox
 - _contentLoaded : bool
 - InitializeComponent() : void
 - IComponentConnector.Connect(int, object) : void
 - DataSourceId : string
 - Field : string
 - Widget1()
 - UpdateControls() : void
 - _caption : string
 - Caption : string
 - Id : string
 - OnActivated() : void
 - OnDeactivated() : void
 - CanConfigure : bool
 - Configure(Window, IList<DataSource>) : bool
 - DataSourceIds : string[]
 - OnRemove(DataSource) : void
 - OnRefresh(DataSource) : void

Overview

- Visual Studio 2012 & .NET 4.5
- ArcGIS Runtime SDK for WPF
- Use templates
- Implement ESRI.ArcGIS.OperationsDashboard interface(s)
- Can make configurable

```
using ESRI.ArcGIS.OperationsDashboard;
using System.ComponentModel.Composition;
using System.Runtime.Serialization;

namespace OpsDashAddin1 {
    [Export("ESRI.ArcGIS.OperationsDashboard.Widget")]
    [ExportMetadata("DisplayName", "Operations Dashboard Widget1")]
    [ExportMetadata("Description", "This is a new widget")]
    [ExportMetadata("ImagePath", "/OpsDash1;component/Images/Widget32.png")]
    [ExportMetadata("DataSourceRequired", true)]
    [DataContract]
    public partial class Widget1 : UserControl, IWidget, IDataSourceConsumer {
```

Building custom widgets

- Dockable window
- Implements IWidget interface
- Most are tied to a data source

IDataSourceConsumer
Interface

Properties

- `DataSourceIds { get; } : string[]`

Methods

- `OnRefresh(DataSource dataSource) : void`
- `OnRemove(DataSource dataSource) : void`

IWidget
Interface

Properties

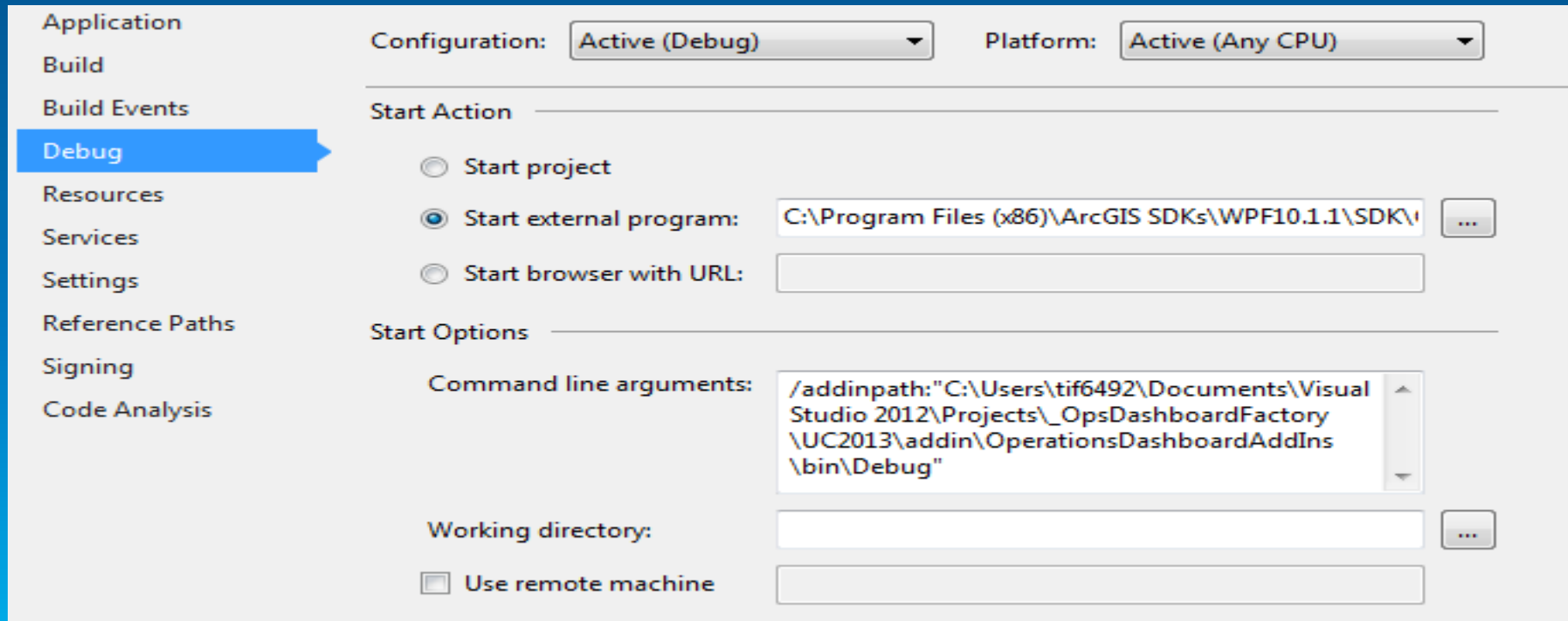
- `CanConfigure { get; } : bool`
- `Caption { get; set; } : string`
- `Id { get; set; } : string`

Methods

- `Configure(Window owner, IList<DataSource> dataSources) : bool`
- `OnActivated() : void`
- `OnDeactivated() : void`

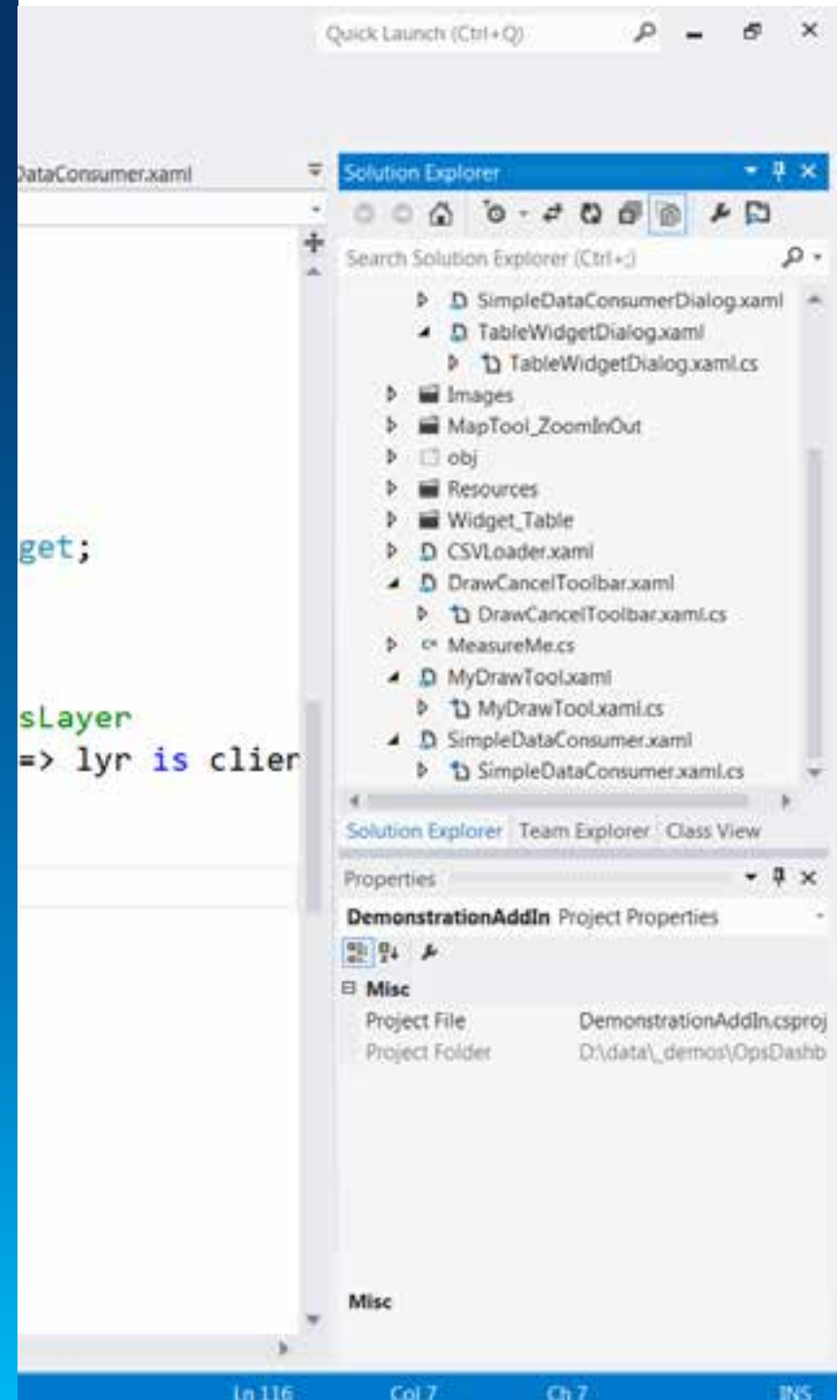
Testing add-ins

- Templates already configured for testing
- Uses a copy of the app included in the SDK



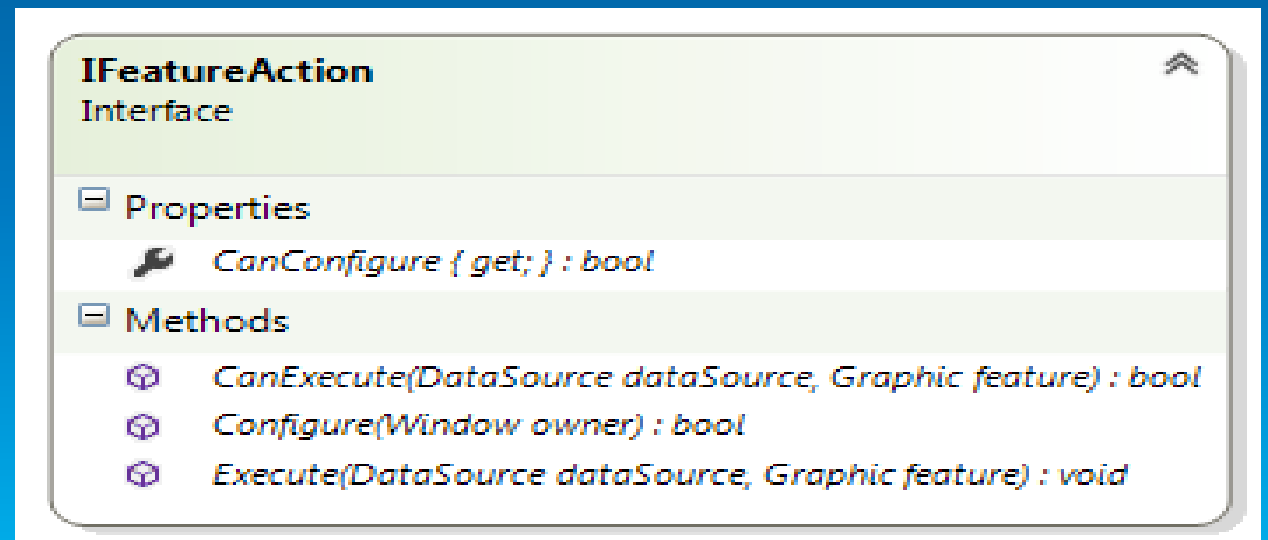
Demo

Building a Widget



Building custom feature actions


- Shown on right-click
- No UI
- Command on a single feature
- Implement IFeatureAction






The screenshot shows the IFeatureAction interface in a development environment. The interface is titled "IFeatureAction" and is labeled as an "Interface". It is organized into two main sections: "Properties" and "Methods".

IFeatureAction
Interface

Properties

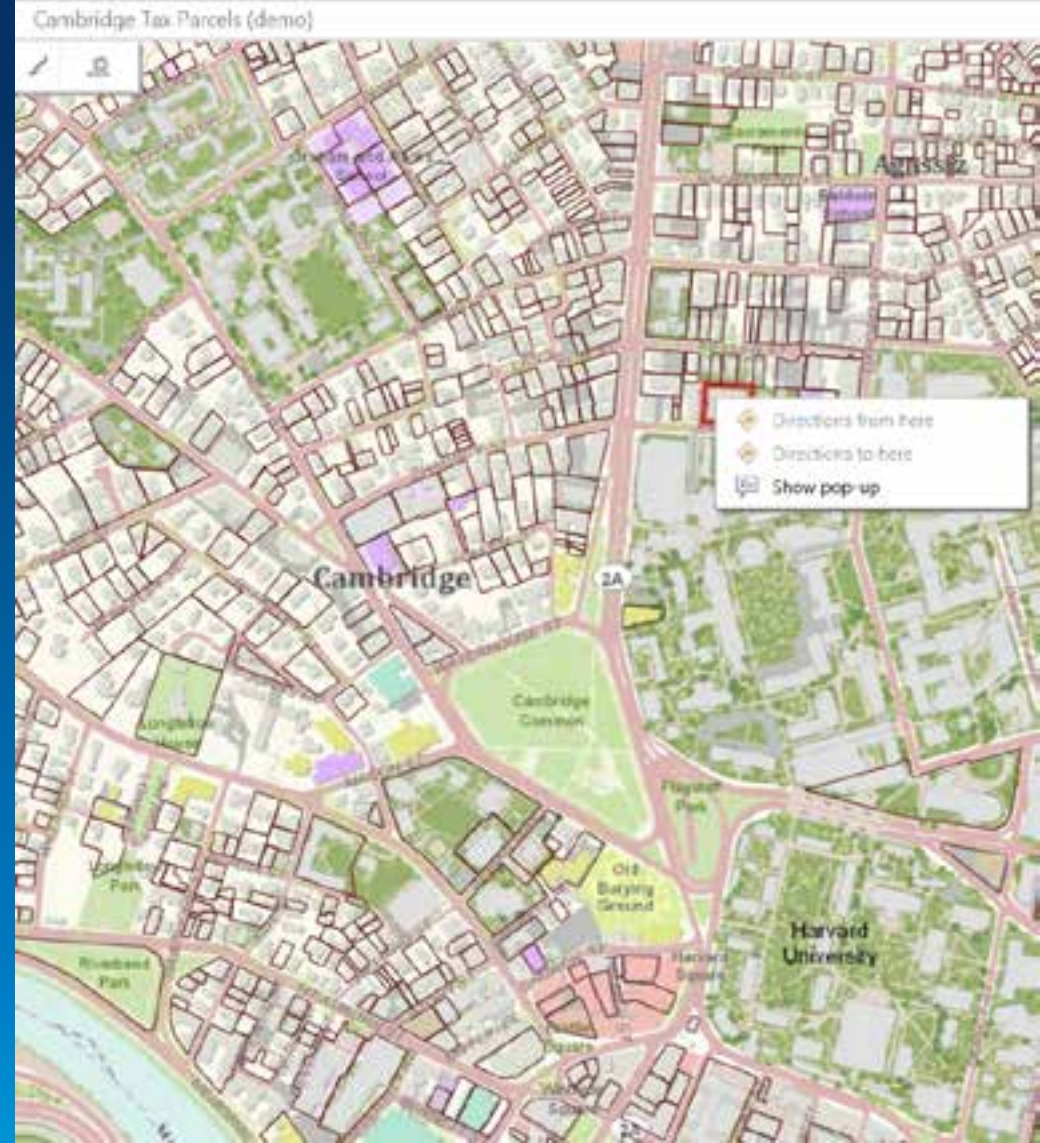
-  *CanConfigure { get; } : bool*

Methods

-  *CanExecute(DataSource dataSource, Graphic feature) : bool*
-  *Configure(Window owner) : bool*
-  *Execute(DataSource dataSource, Graphic feature) : void*

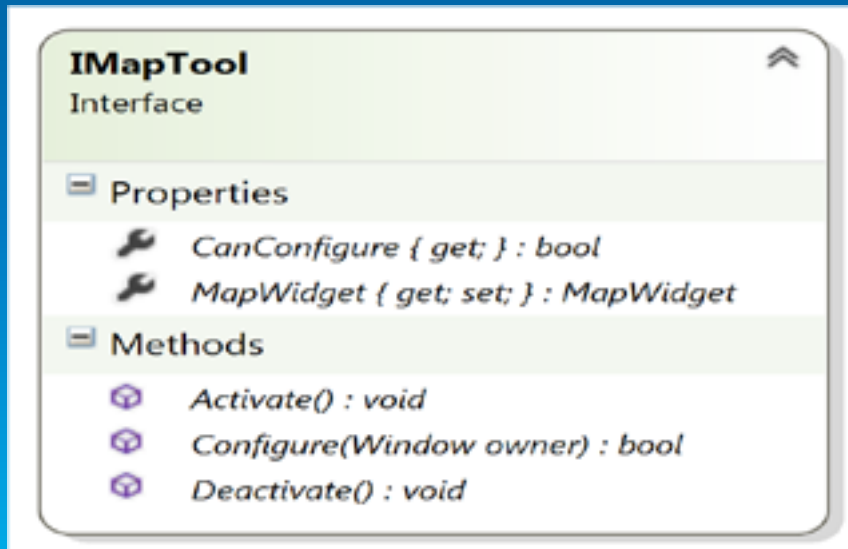
Demo

Demo: Building a Feature Action



Building custom map tools

- Appear on map toolbars
- Use to interact with the map
- Can use a temporary toolbar
- IMapTool & IMapToolbar interfaces



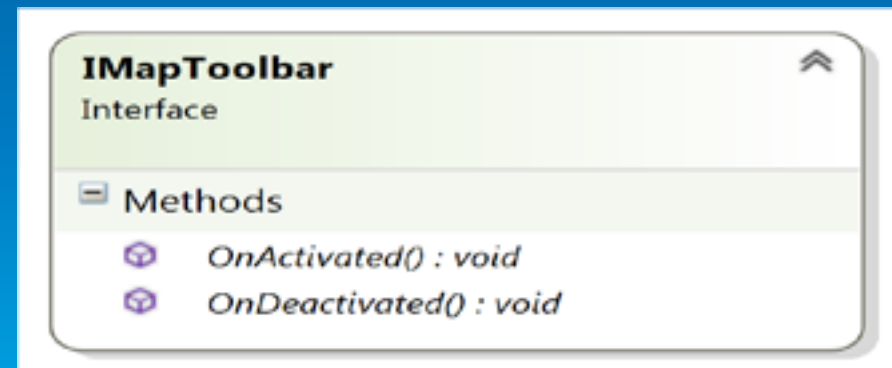
IMapTool
Interface

Properties

- CanConfigure { get; } : bool*
- MapWidget { get; set; } : MapWidget*

Methods

- Activate() : void*
- Configure(Window owner) : bool*
- Deactivate() : void*



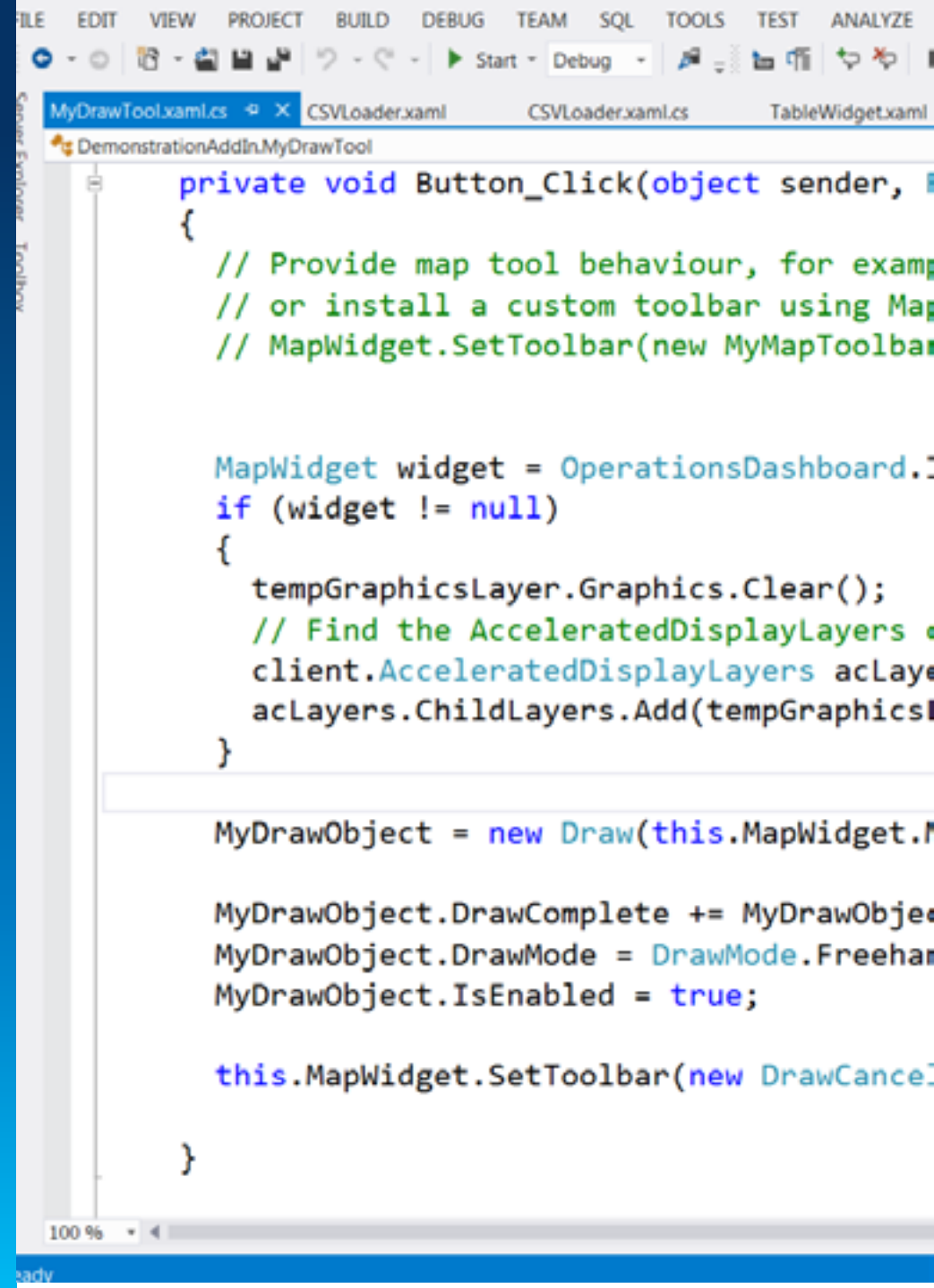
IMapToolbar
Interface

Methods

- OnActivated() : void*
- OnDeactivated() : void*

Demo

Building custom map tools



```
private void Button_Click(object sender, EventArgs e)
{
    // Provide map tool behaviour, for example
    // or install a custom toolbar using MapWidget.Toolbar
    // MapWidget.SetToolbar(new MyMapToolbar());

    MapWidget widget = OperationsDashboard.MapWidget;
    if (widget != null)
    {
        tempGraphicsLayer.Graphics.Clear();
        // Find the AcceleratedDisplayLayers collection
        client.AcceleratedDisplayLayers acLayers =
            widget.Client.AcceleratedDisplayLayers;
        acLayers.ChildLayers.Add(tempGraphicsLayer);
    }

    MyDrawObject = new Draw(this.MapWidget.Map);

    MyDrawObject.DrawComplete += MyDrawObject_DrawComplete;
    MyDrawObject.DrawMode = DrawMode.Freehand;
    MyDrawObject.IsEnabled = true;

    this.MapWidget.SetToolbar(new DrawCancelToolbar());
}
}
```

Sharing add-ins

Table, SearchNearby, ZoomIn



Dashboard Addins for UC2013

Operations Dashboard Add In by tif_nitro
Last Modified: July 5, 2013

☆☆☆☆☆ (0 ratings, 9 views)

 Facebook  Twitter

OPEN

Description

Dashboard Addins for UC2013, including: Table (Widget), Search nearby (F

Access and Use Constraints

provided as is

Properties

Tags	Table, SearchNearby, ZoomInOut
Credits	
Size	26 KB
Extent	

Comments (0)

Comments have been disabled on this item.

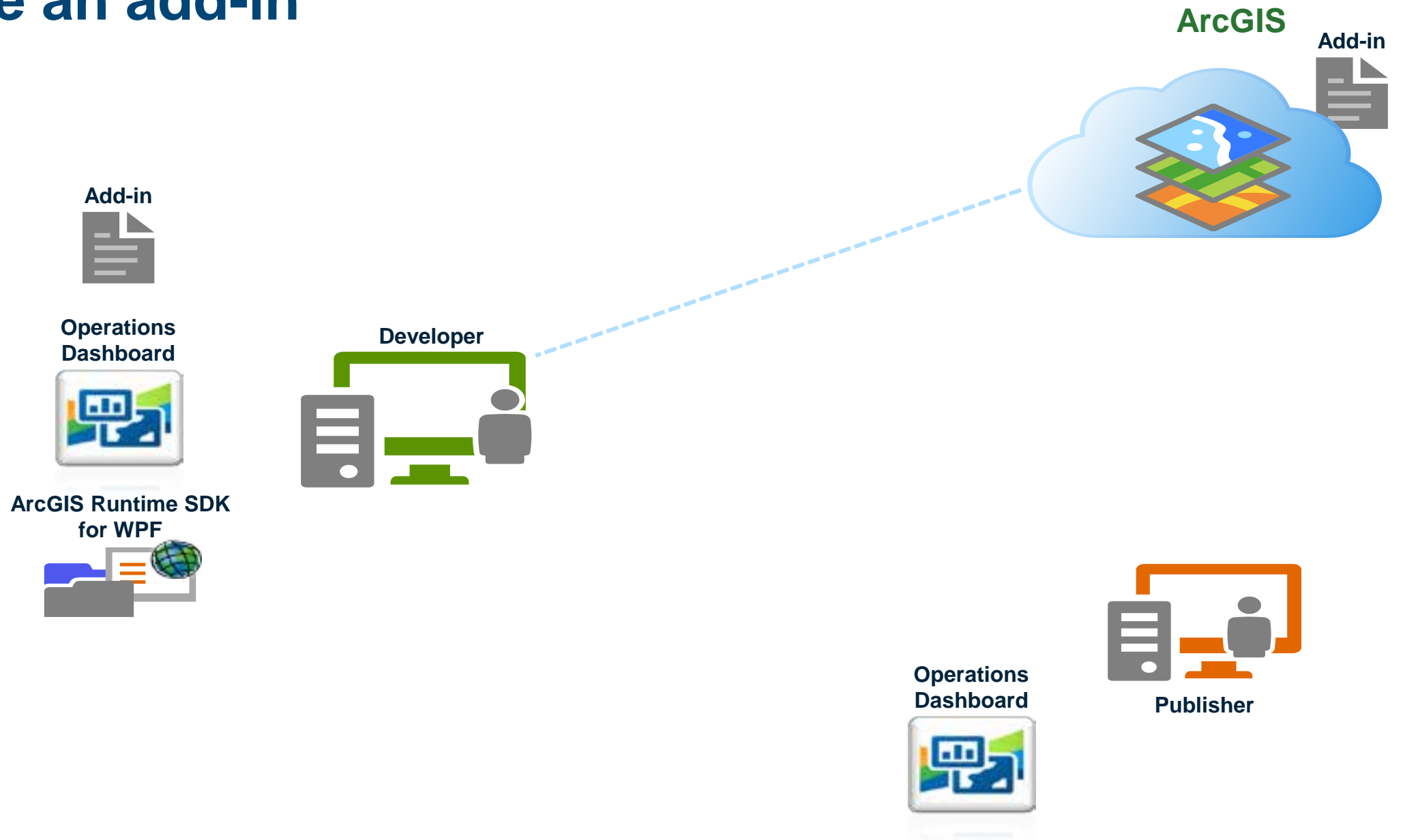
Sharing add-ins – Deploying

- **Create the .opdashboardAddin file**
- **Upload to ArcGIS Online or Portal for ArcGIS**
- Update it à Upload again.
- **Automatically available to users**

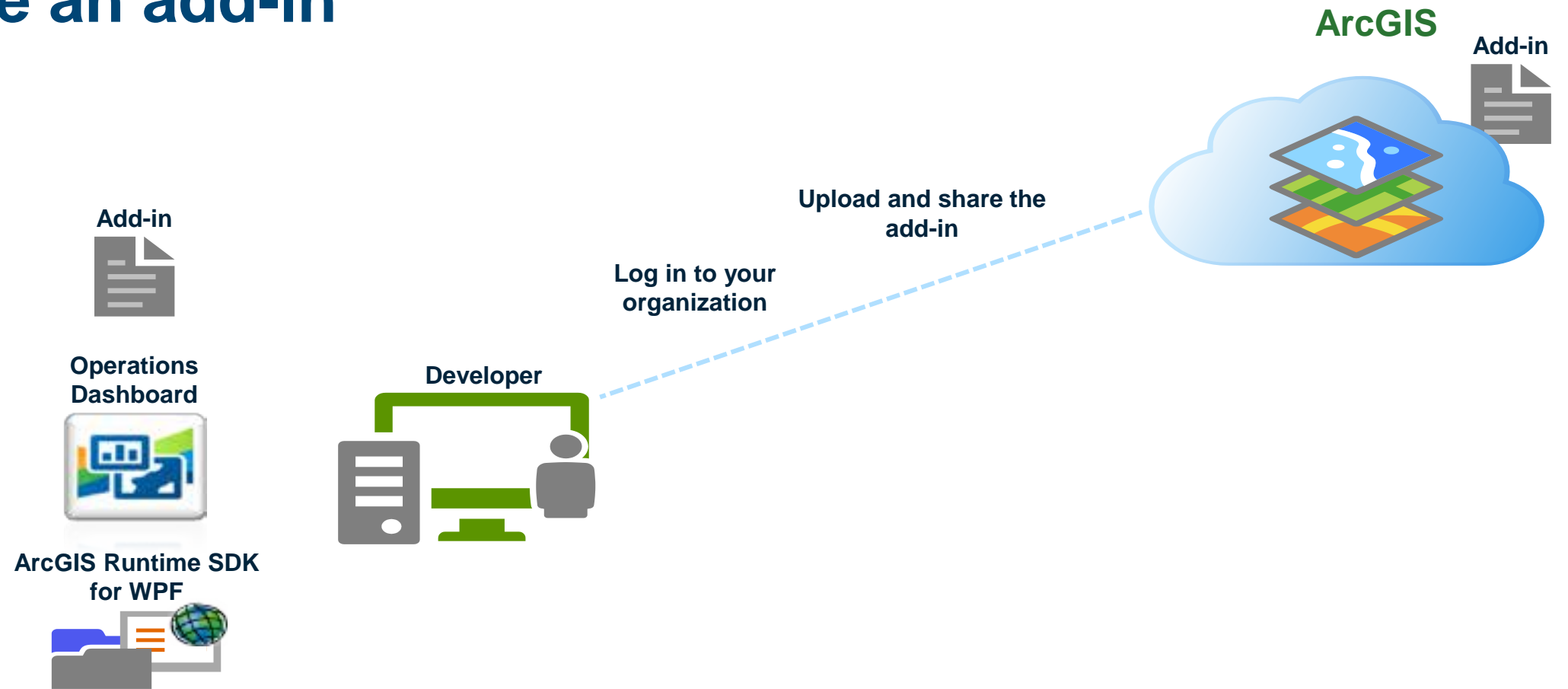
Share an add-in



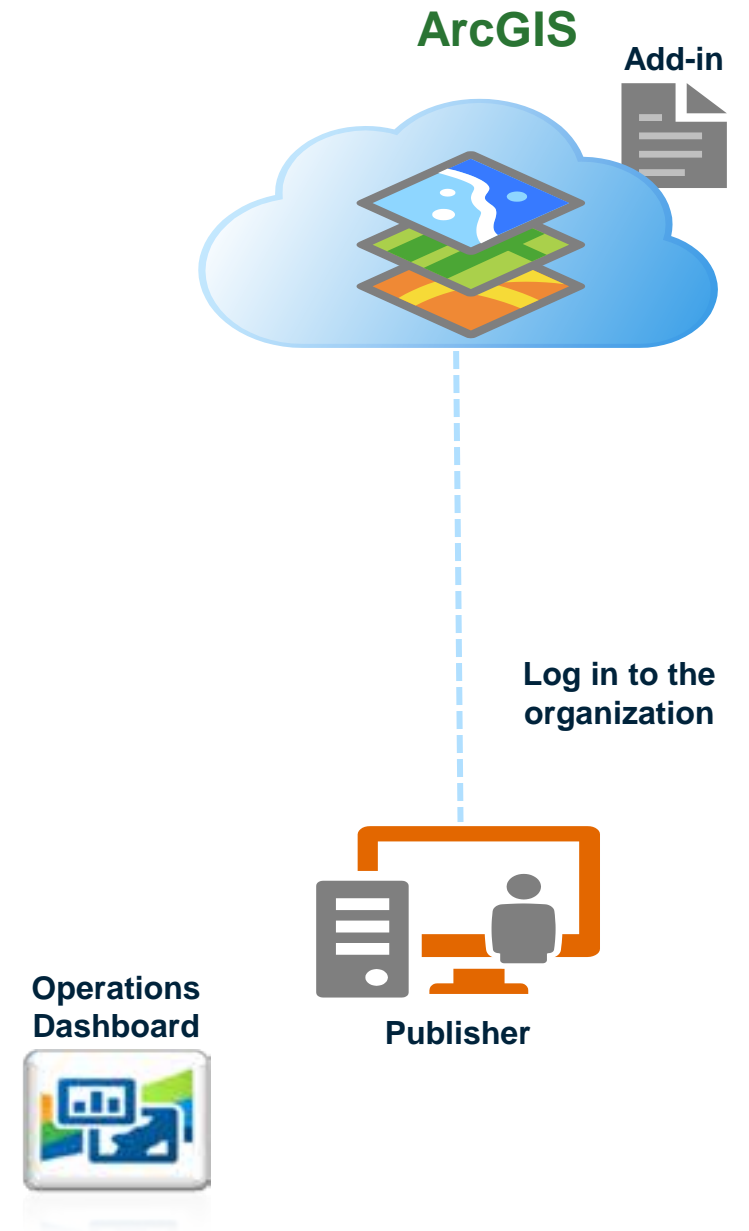
Share an add-in



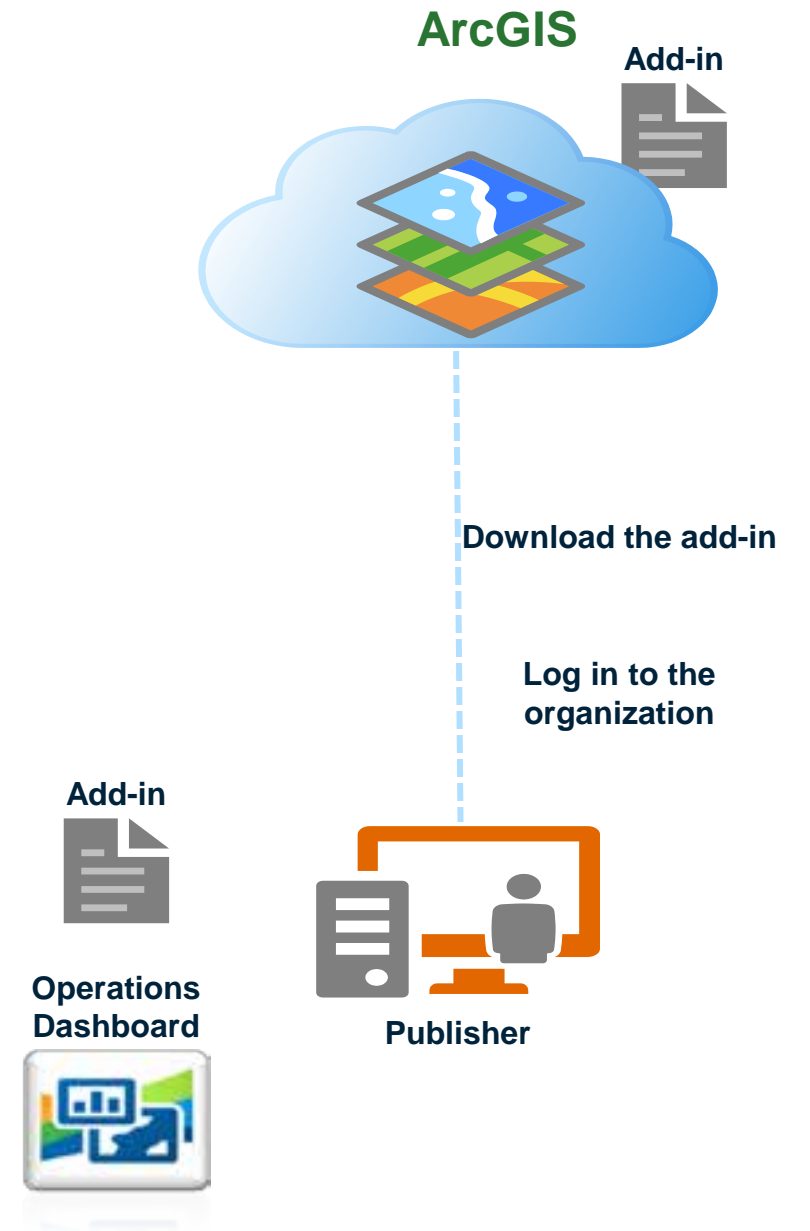
Share an add-in



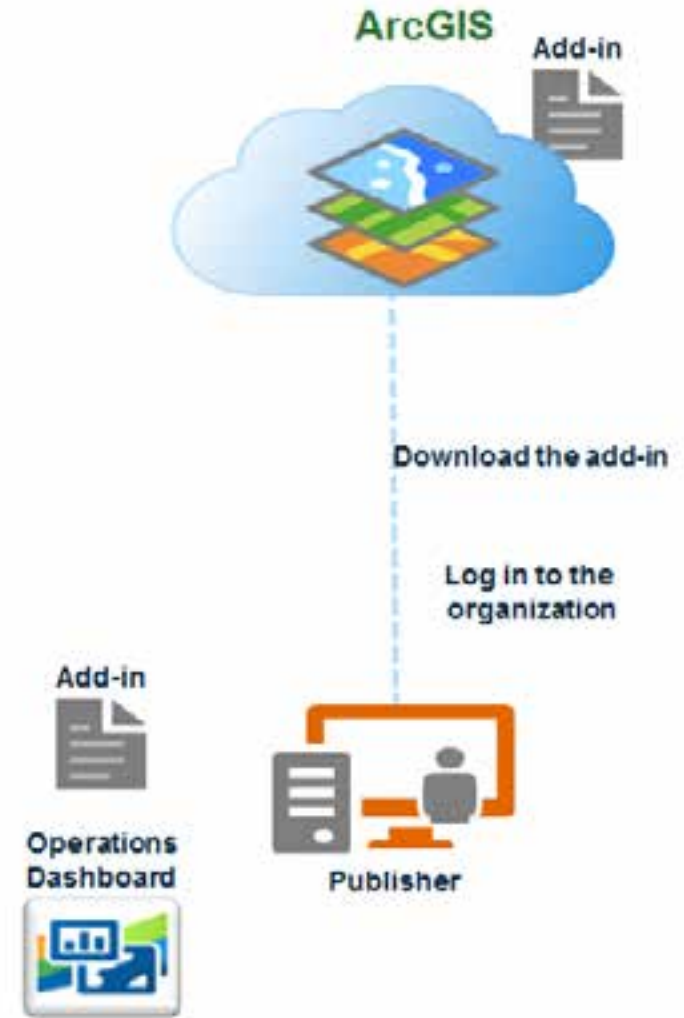
Share an add-in



Share an add-in

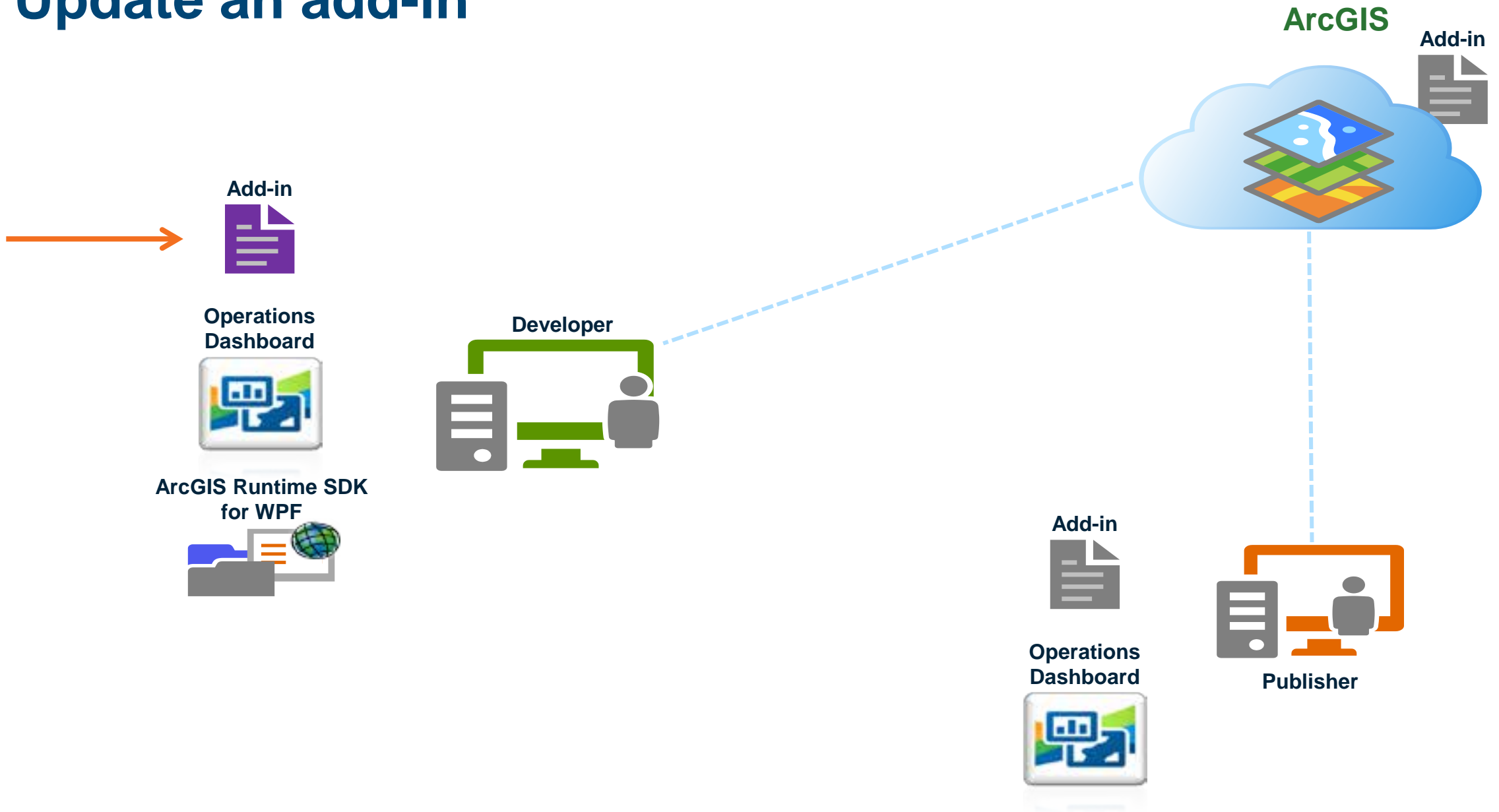


Share an Add-In

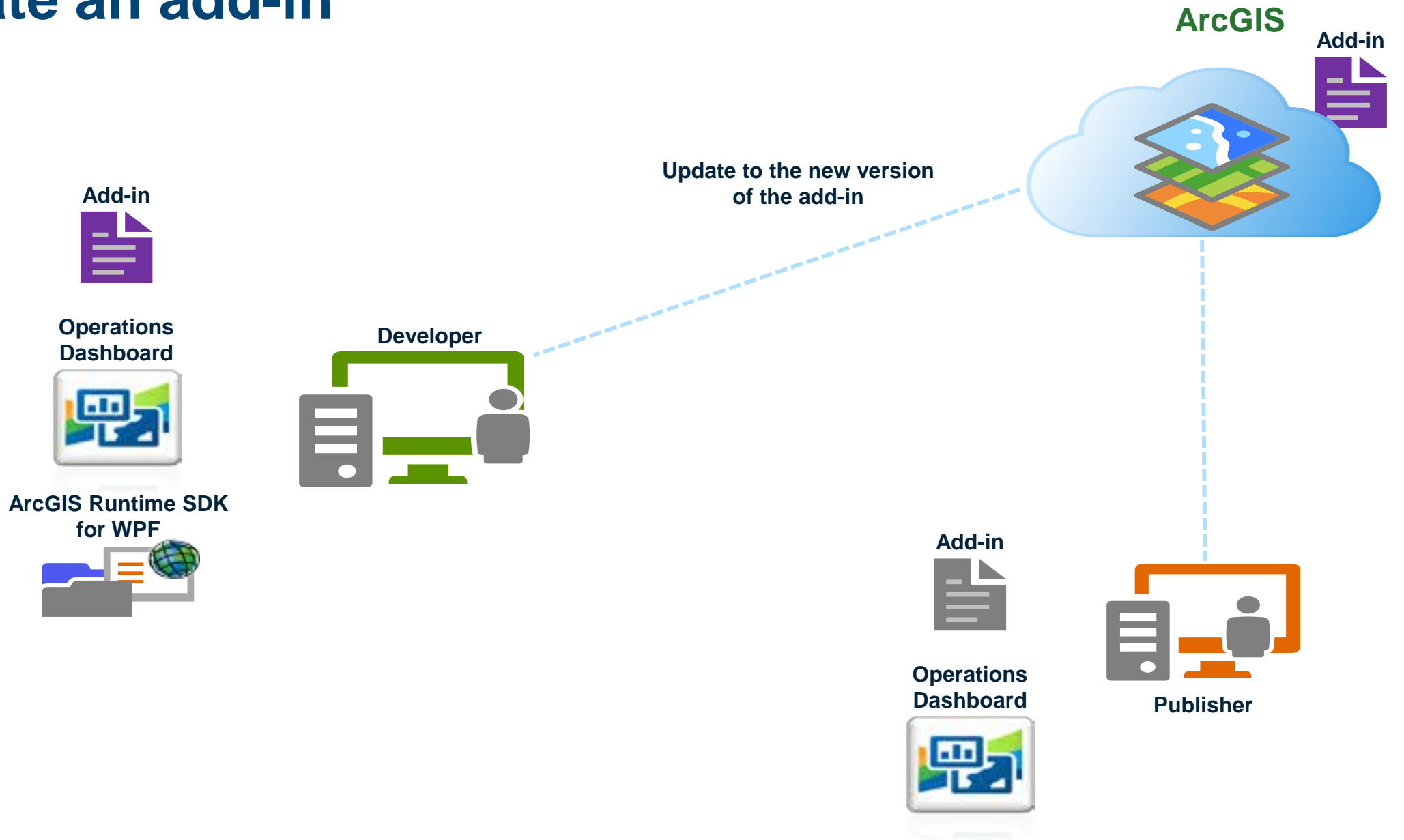


Updating an Add-In

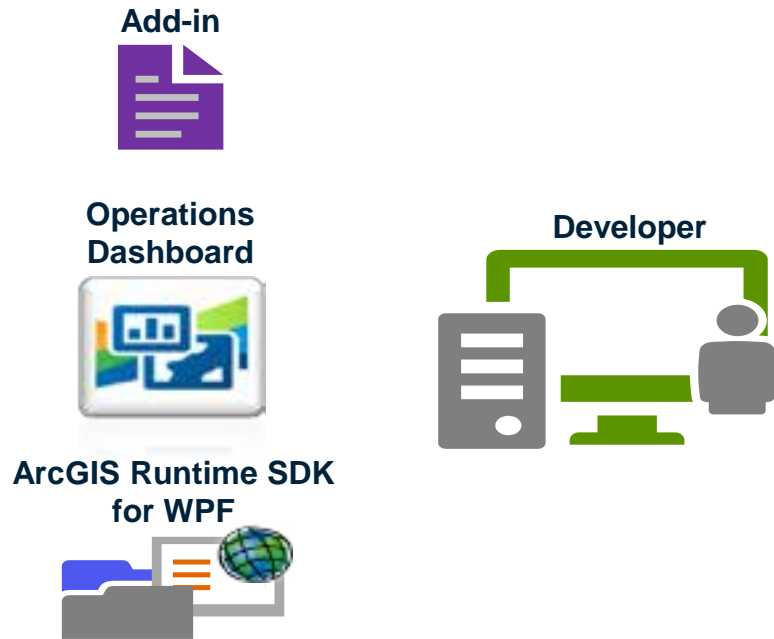
Update an add-in



Update an add-in

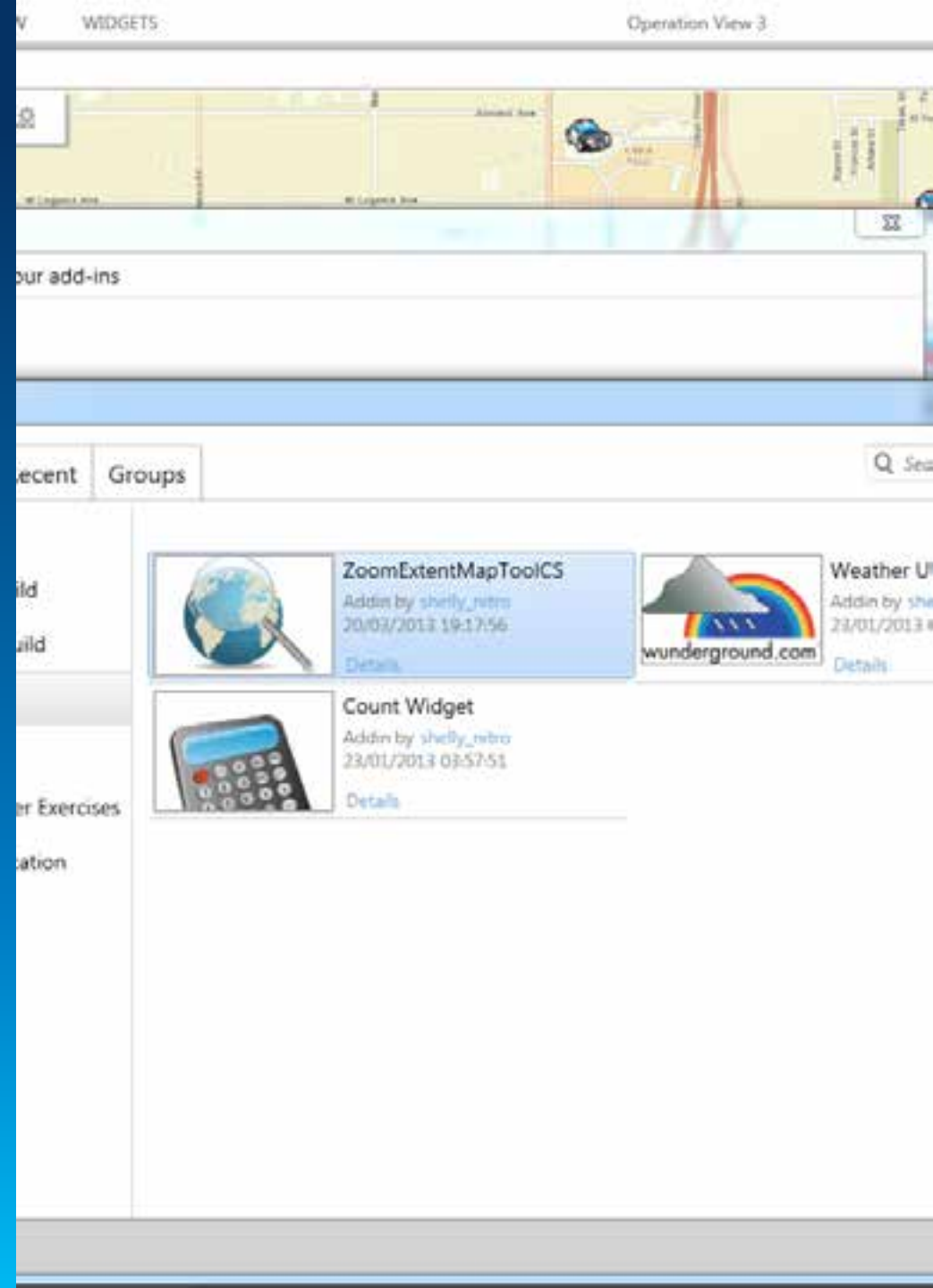


Update an add-in



Demo

Updating an Add-In



Summary

- **Operations Dashboard Behind the Scenes**
- **Operational Views**
- **Types of tools that can be built within AddIns**
- **Sharing and Updating AddIns**

Thank you.

- **Questions?**



Understanding our world.