Welcome

[YOUR TRAINER THIS SESSION]

DevOps Engineer
Data Exploration

Klaas Dewitte

klaas.dewitte@skyline.be

Creating your own – GQI → Data source

Agenda

[WHAT YOU WILL SEE THIS SESSION]

Advanced track

1. Core concepts
2. Creating your own data source
3. Harnessing the logger
4. Taking in input
5. Supporting real-time updates
6. Getting data from DataMiner

Before we start

Some new things

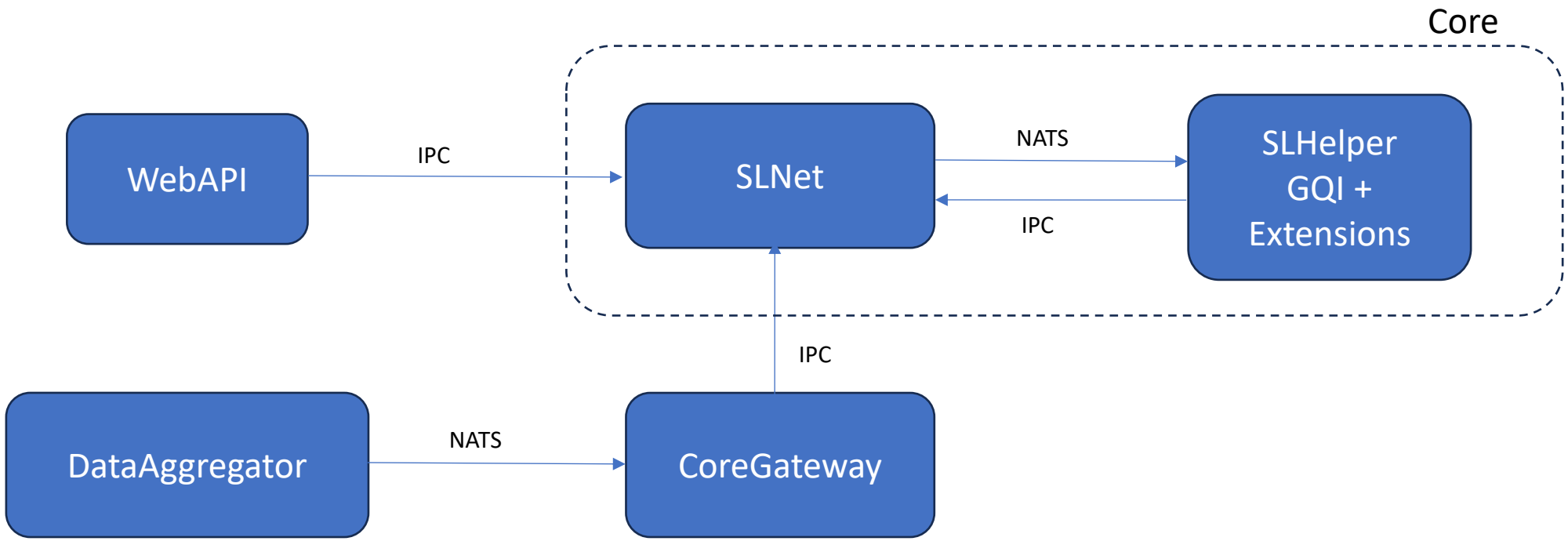
What's new?

GQI as DataMiner Extension Module (DxM)

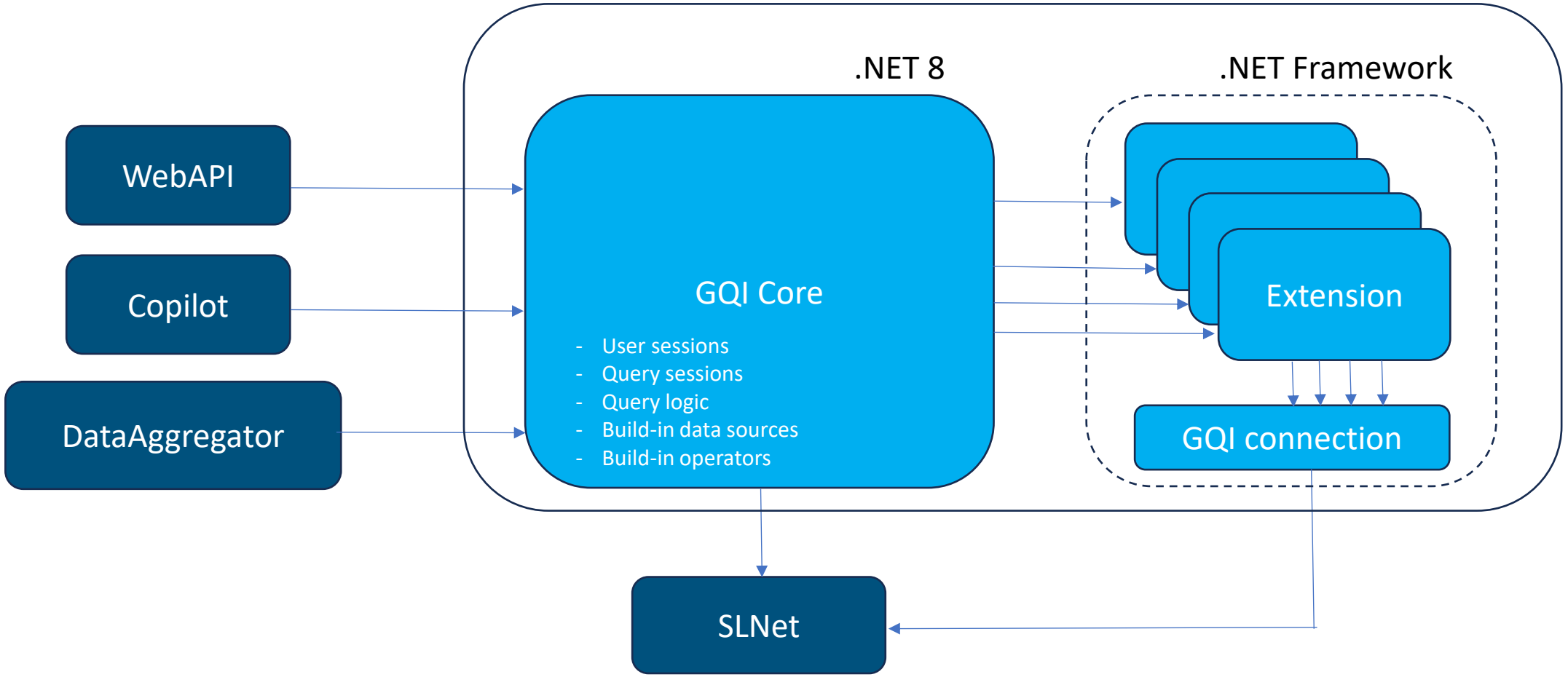
- Independently upgradeable
  - Included in web-only upgrade packages
  - No downtime for DataMiner agents
- Performance enhancements
- Extensions run inside separate processes:
  - Improved dependency resolving avoids version conflicts of NuGet/DLLs
  - Resource usage visible per extension
  - An extension can be killed separately without ending others/the core GQI process

Architectural Changes

Situation before



Ad hoc data sources in the GQI DxM



Recap on GQI extensions

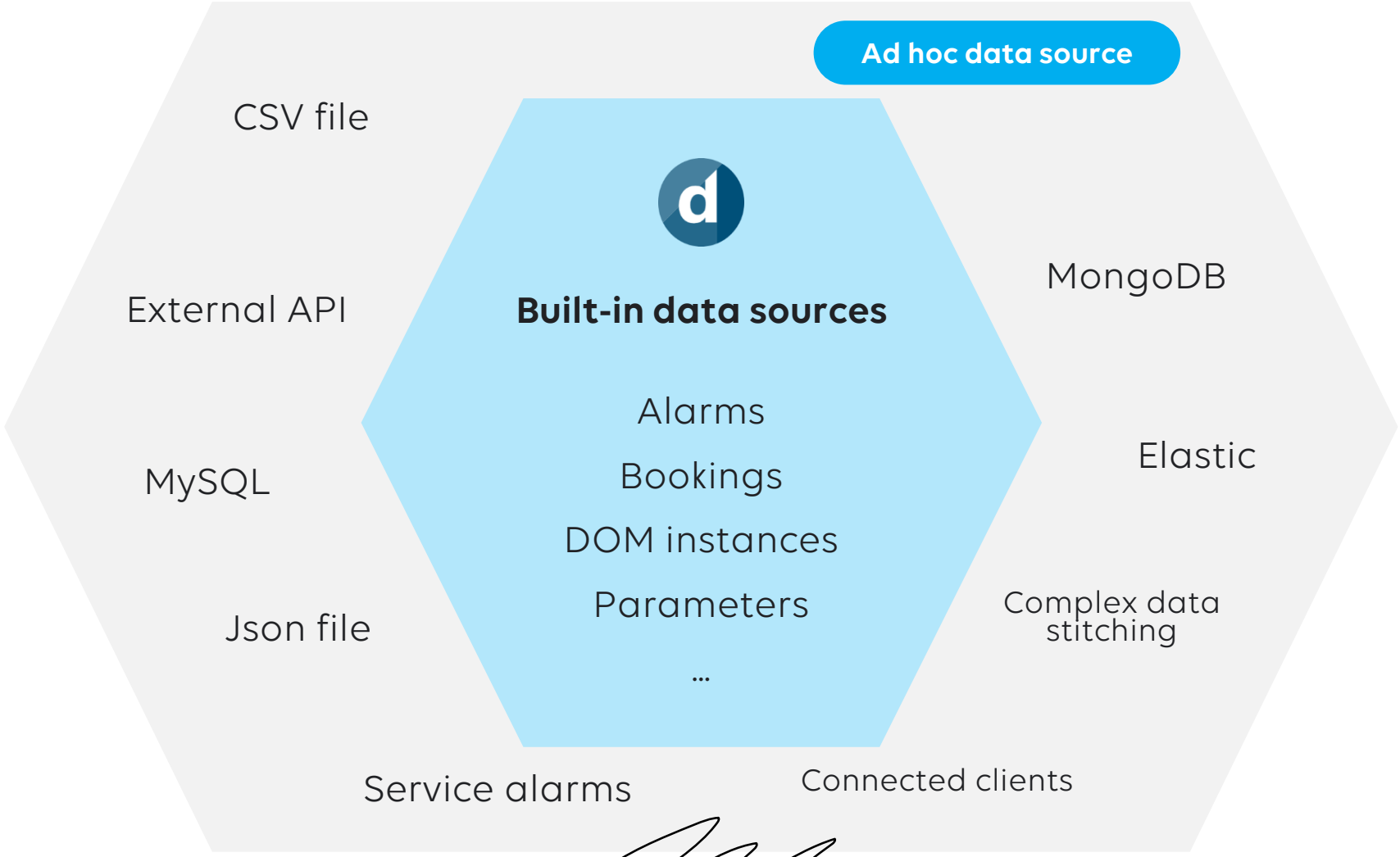
Why should I care

Ad hoc data source

Custom operator

- C# code
- Compiled as library (DLL) by Automation module
- Source code visible in DataMiner Cube

Why extend GQI data sources?



When creating GQI data sources?

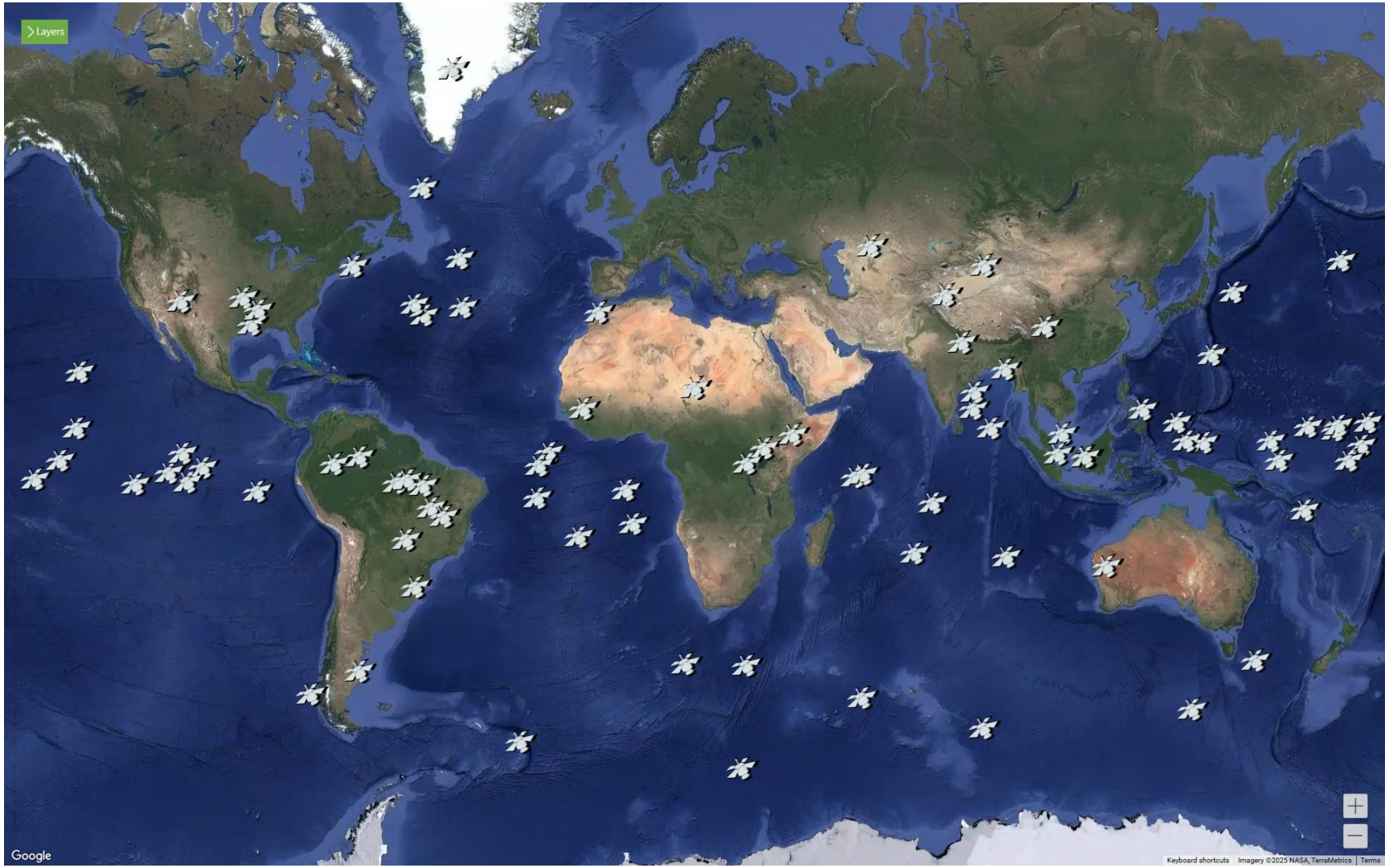
Less effort to get your data in Dashboards and Low-code apps

	DataMiner connector	GQI ad-hoc data source
Trending	Yes	No
Alarming	Yes	No
Automation	Yes	No
Correlation	Yes	No
DataMiner Cube	Yes	Requires embedding a dashboard or low-code app
Dashboards	Yes	Yes
Low-code apps	Yes	Yes
Ad hoc	No	Yes

Creating your own data source

And so it begins

What we'll be creating



Package

Empower - Create your own GQI data source

Browse catalog

SLC-GQIDS-SatellitesDemo

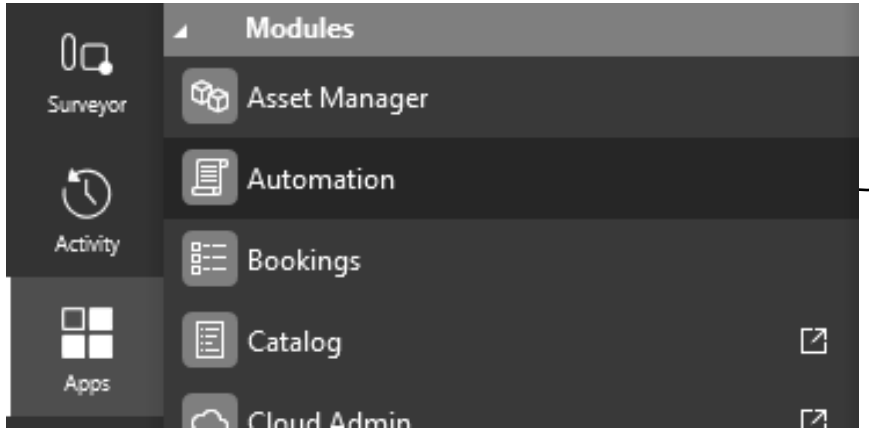
Creating your own data source

Prerequisites

- DataMiner Cube

- Recommended:
  - Visual Studio
  - DataMiner Integration Studio (DIS)
    https://marketplace.visualstudio.com/items?itemName=skyline-communications.DataMinerIntegrationStudio

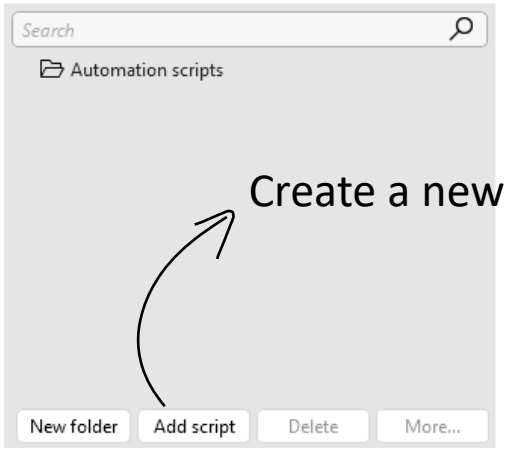
Where to find your data sources



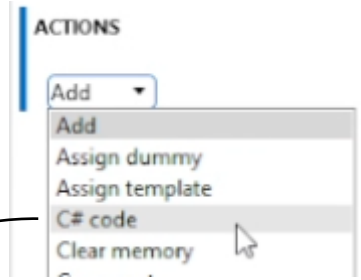
Part of the automation module in Cube

How to create

automation scripts memory files

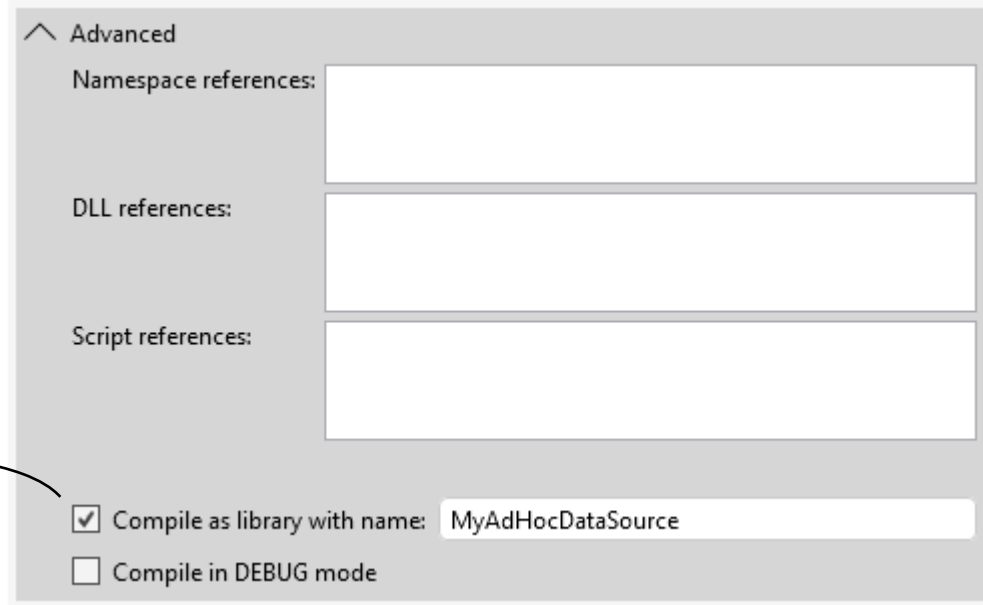


Create a new automation script



Add a C# action

Add some C# code

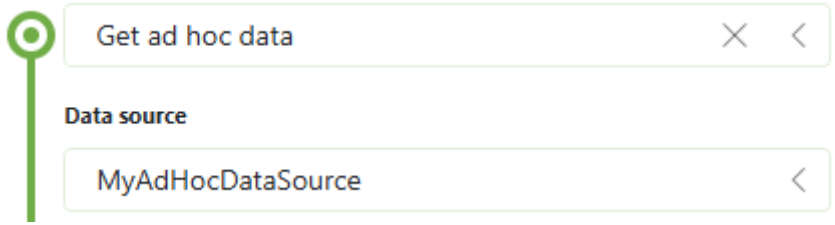


Compile as library

How to create

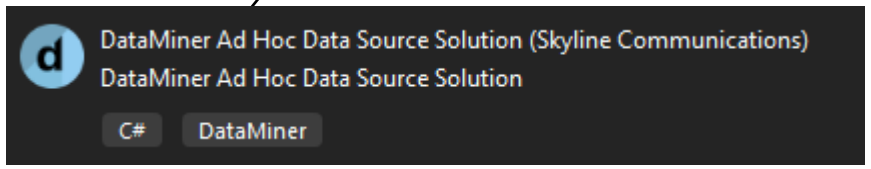
Save the automation script

The data source can now be found in dashboards and low code apps

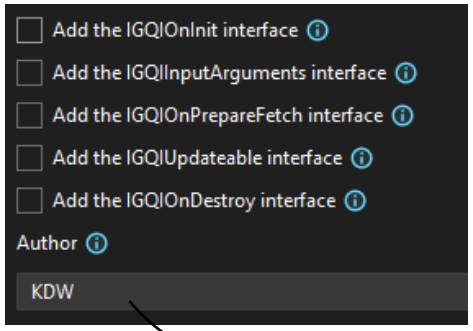


Creating with DIS

Starting from Visual Studio package

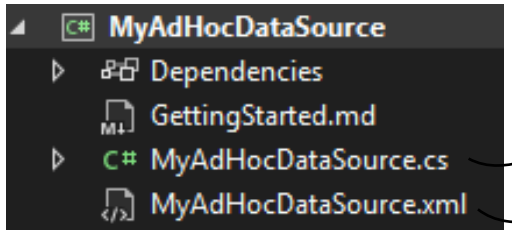


Give your data source a name



Add an Author

Creating with DIS



Ad hoc data source code

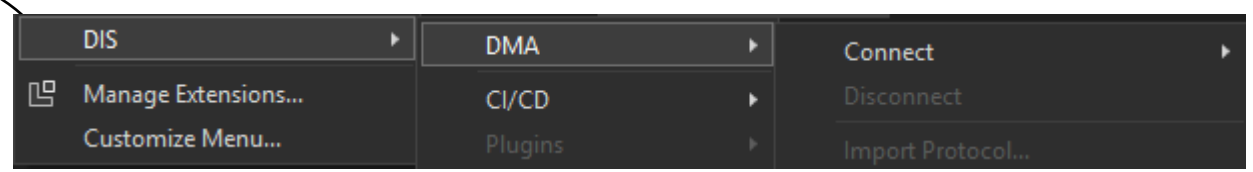
Automation script definition

A basic data source is provided

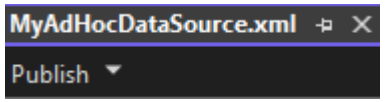
```
using System;
using Skyline.DataMiner.Analytics.GenericInterface;

namespace MyAdHocDataSource
{
    /// <summary>
    /// Represents a data source.
    /// See: https://aka.dataminer.services/gqi-external-data-source for a complete example.
    /// </summary>
    [GQIMetaData(Name = "MyAdHocDataSource")]
    0 references
    public sealed class MyAdHocDataSource : IGQIDataSource
    {
        0 references
        public GQIColumn[] GetColumns()
        {
            // Define data source columns
            // See: https://aka.dataminer.services/igqidatasource-getcolumns
            return Array.Empty<GQIColumn>();
        }

        0 references
        public GQIPage GetNextPage(GetNextPageInputArgs args)
        {
            // Define data source rows
            // See: https://aka.dataminer.services/igqidatasource-getnextpage
            return new GQIPage(Array.Empty<GQIRow>())
            {
                HasNextPage = false,
            };
        }
    }
}
```

Publishing to a DMA

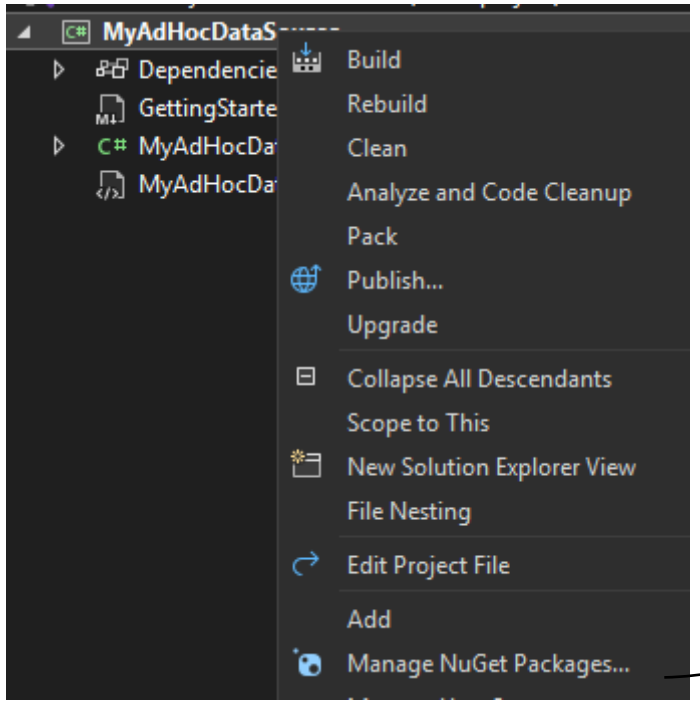
Connect to your agent



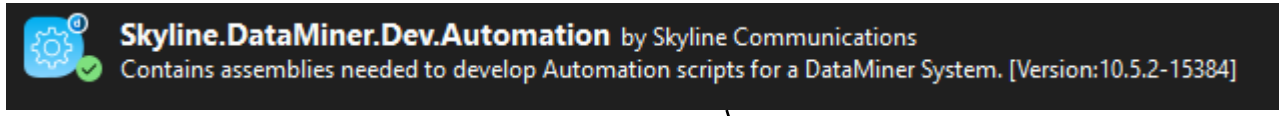
And publish with ease



(Optional) Upgrade package to latest

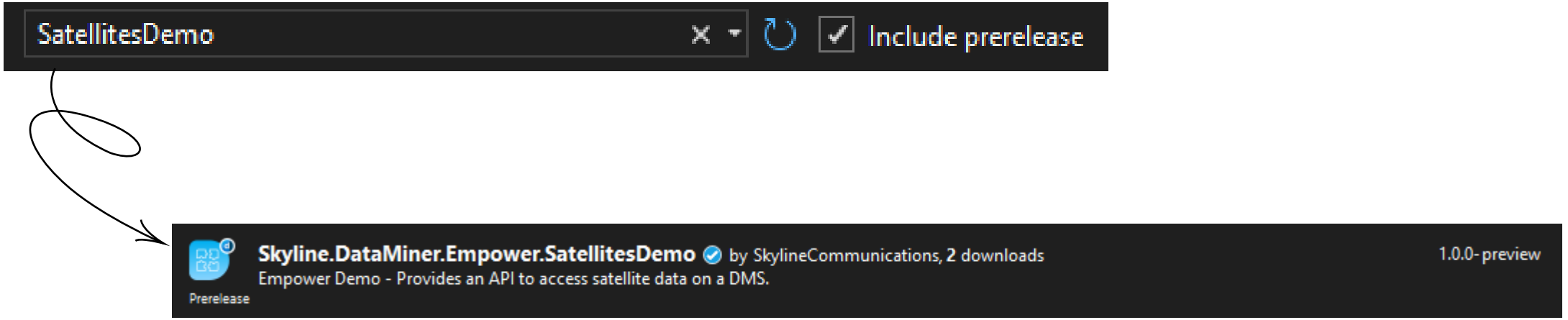


Manage NuGet Packages…



Update to newest version

Demo package



Let's create!

A very basic Ad Hoc data source

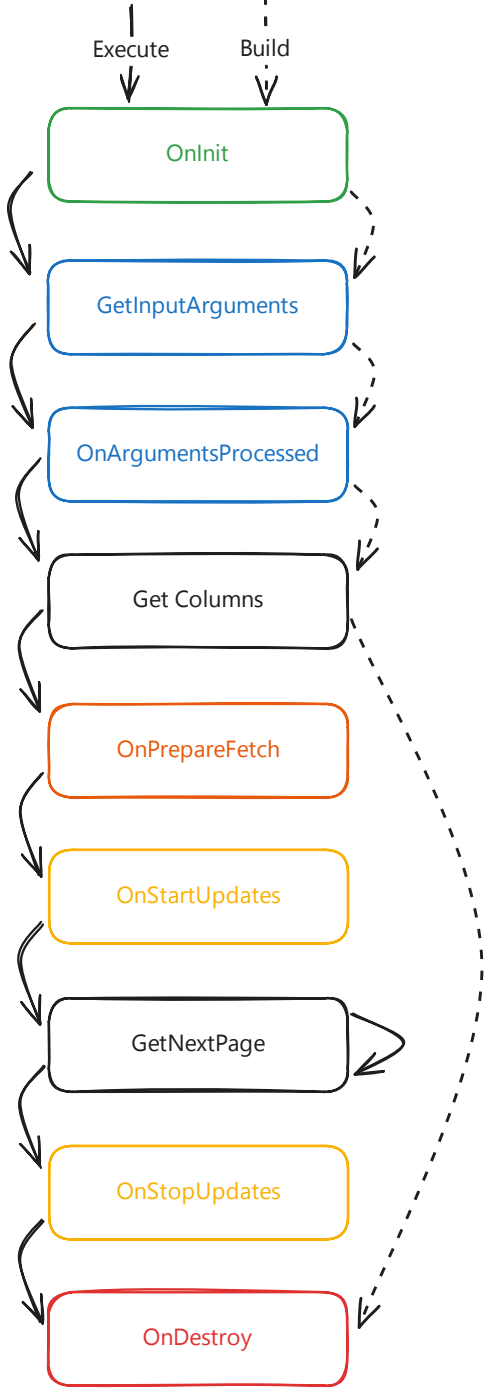
What is an Ad-hoc data source

Expose ANY data from any place, just in time

What is an Ad hoc data source

The full life cycle

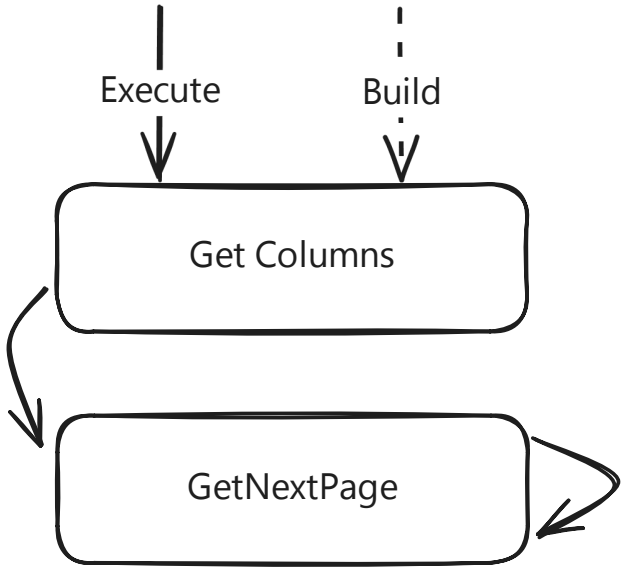
- Each step can be implemented
- Defined by a C# class
- An instance is created for every query execute
- The C# class implements interfaces



The building blocks

The lifecycle

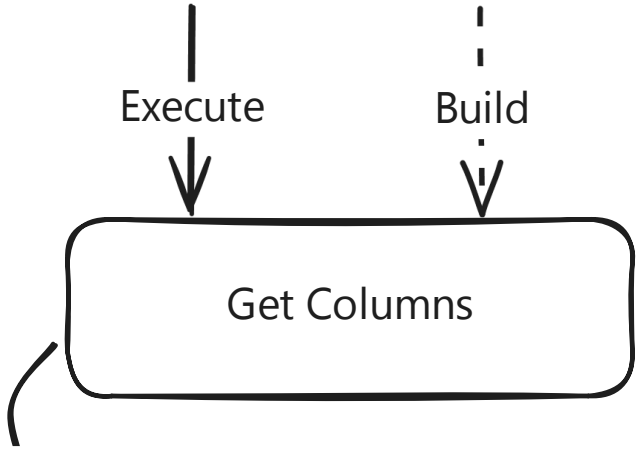
- Only required blocks to create a data source
- C# interface: IGQIDataSource



Get Columns

The lifecycle

- Provide what will be present in the data source
- In the form of an array of GQIColumns

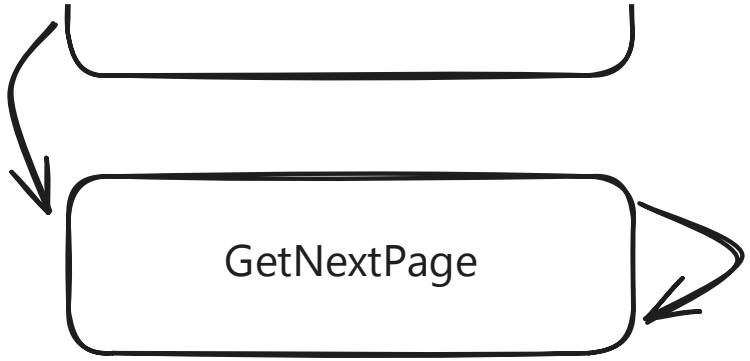


Let's define some columns!

Get Next Page

The lifecycle

- Provide the data
- In the form of a GQIPage

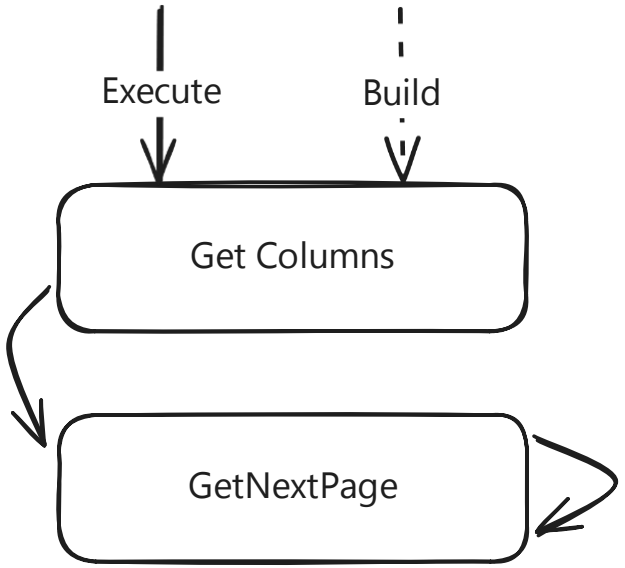


We now have data!

Harnessing the logger

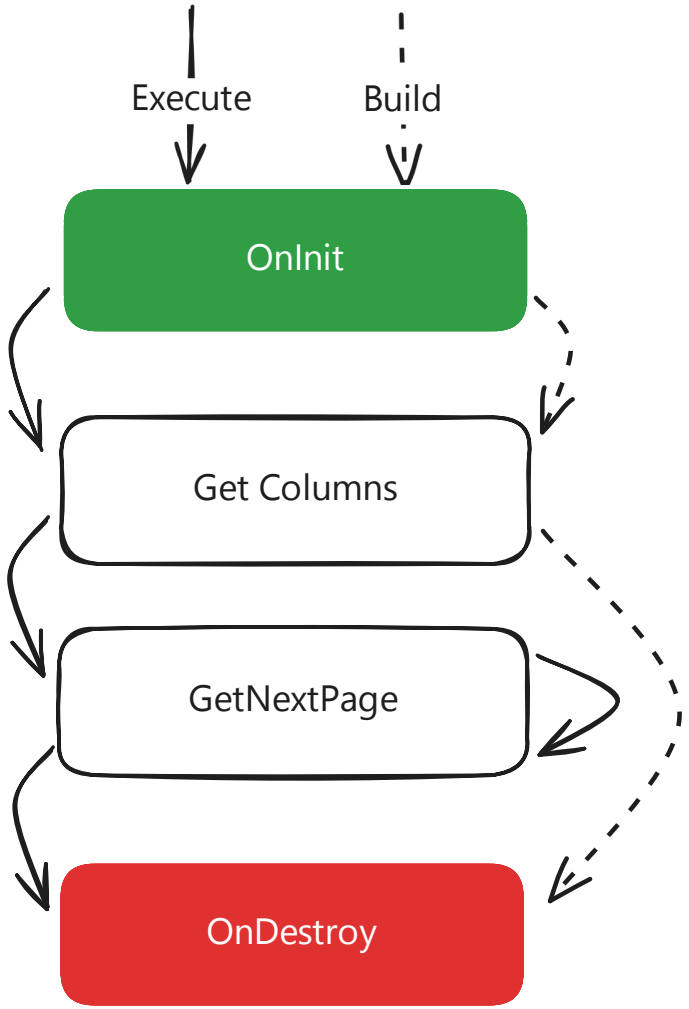
Let add some more blocks

- What we had before



Let add some more blocks

- What we had before

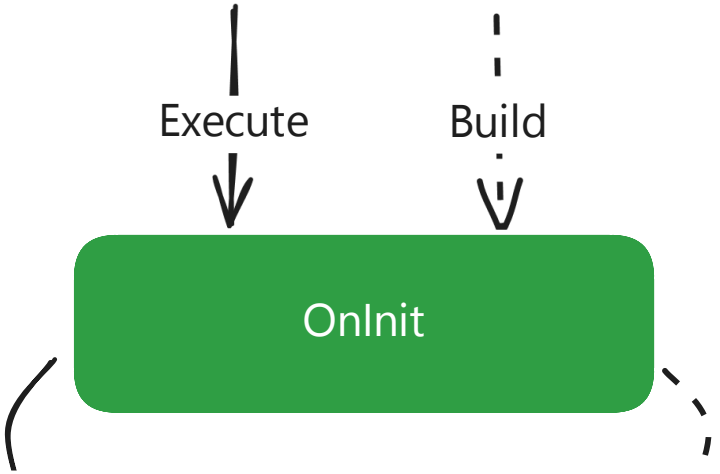


OnInit

- Is invoked once for every instance of a data source
- Provided from C# interface: IGQIOnInit

- Init is the first method that is invoked

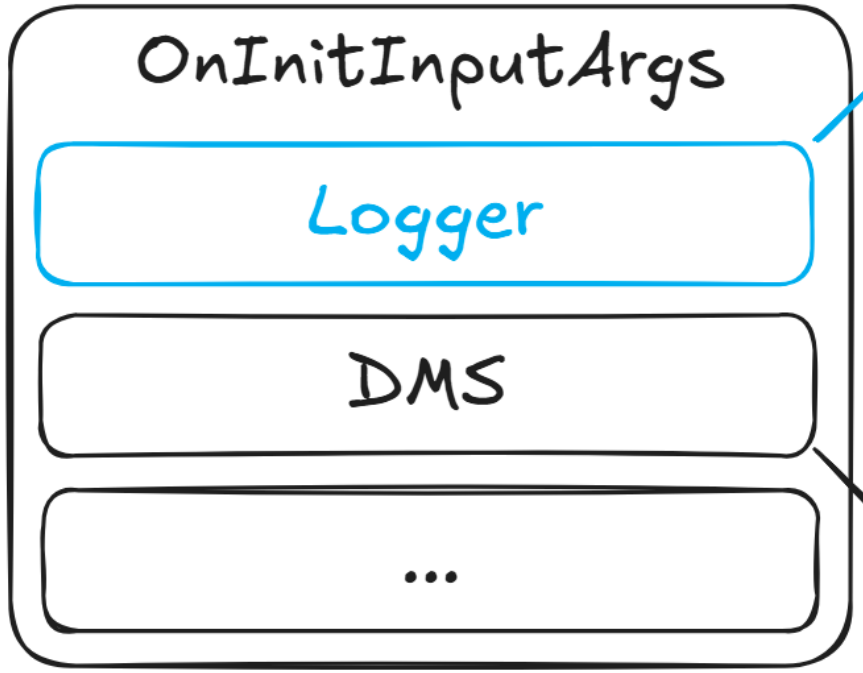
- Provides OnInitInputArgs



Let's implement IGQIOnInit

OnInit

OnInitInputArgs



Allows creators to log custom messages about their data source

Allows creators to access their DataMiner system

Let's log!

The logger

- Received in OnInit

DOCS

```
{
    "Serilog": {
        "MinimumLevel": {
            "Default": "Error",
            "Override": {
                "Microsoft": "Information"
            }
        }
    },
    "GQIOptions": {
        "Extensions": {
            "Logging": {
                "MinimumLogLevel": "Error"
            }
        }
    }
}
```

Fully configurable
(appsettings.custom.json)

Stored in its own location
One file per library

OnDestroy

Clean up your mess

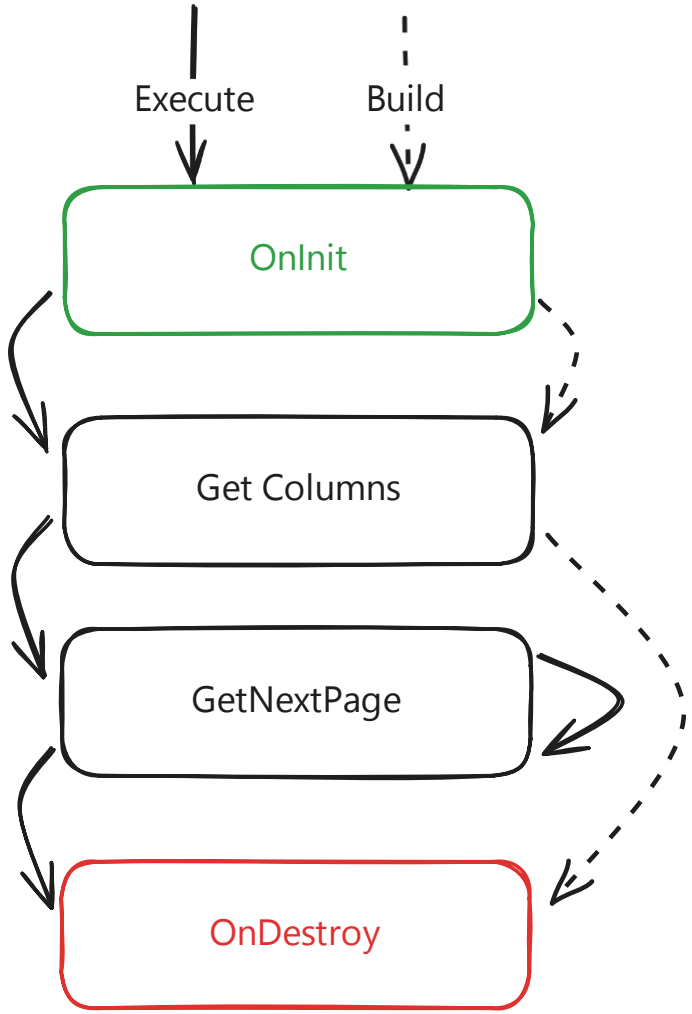
- Is invoked once for every instance of a data source
- Provided from C# interface: IGQIOnDestroy

- Destroy is the last method that is invoked

- Execute when the instance will no longer be used
- Can be used to clean up resources

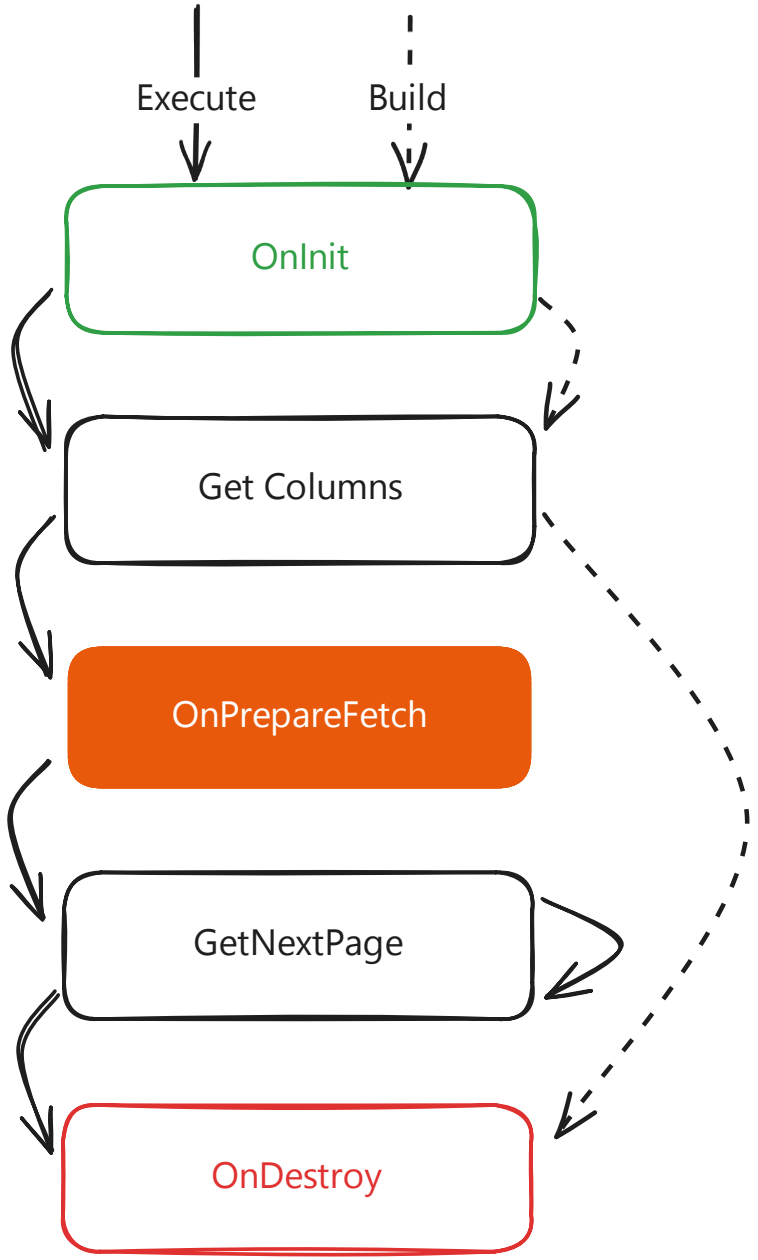
Improving remote request

- Another block is needed



OnPrepareFetch

- Another block is needed

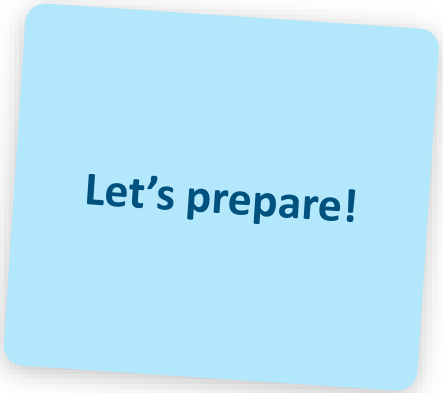


OnPrepareFetch

Let's get ready

- Is invoked once for every instance of a data source
- Right before starting the fetch data
- Provided from C# interface: IGQIOnPrepareFetch

- Execute logic to fetch data that only needs to happen once for all pages

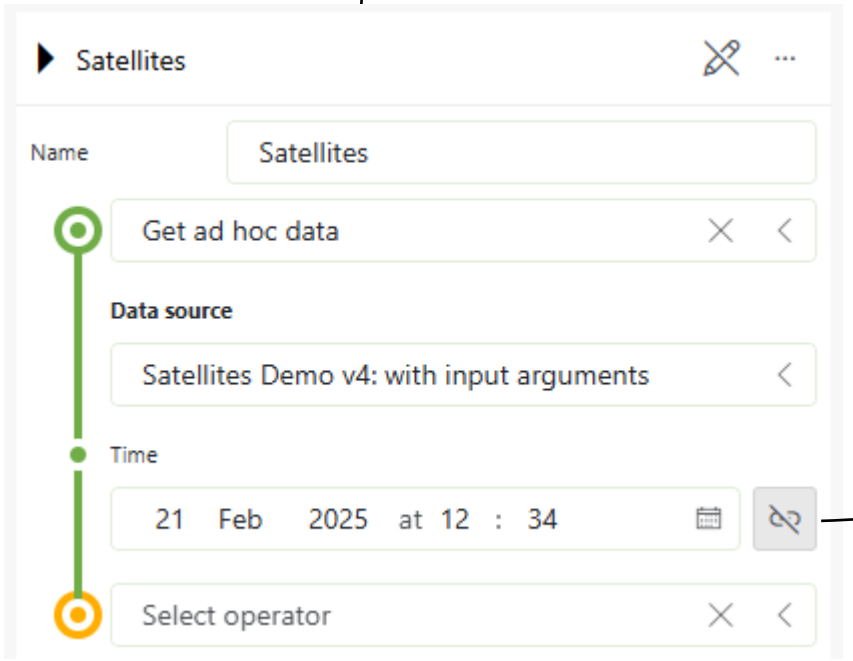


Taking in input

Because not all users want the same

Dynamically modifying the data source

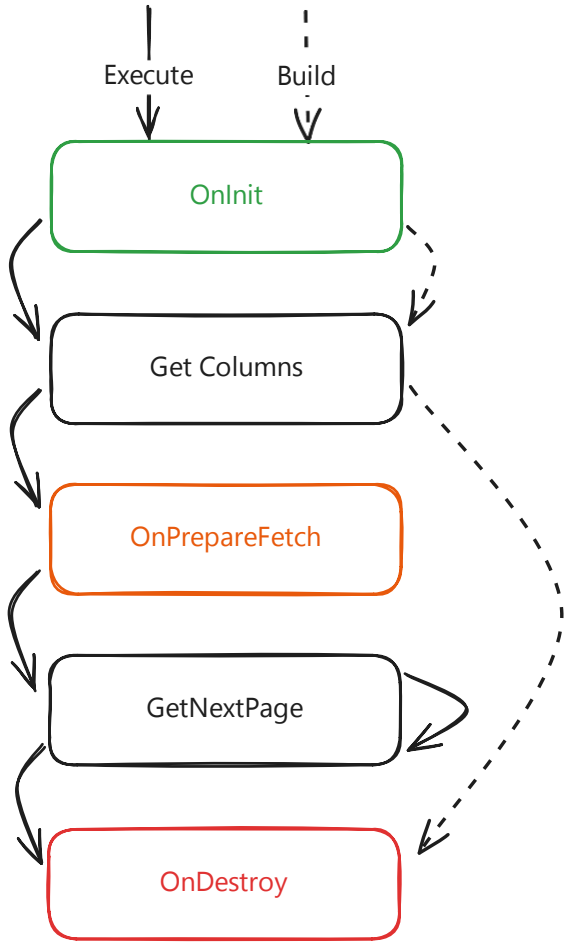
Let users interact with your data source



Link to any data

The life cycle

Reminder

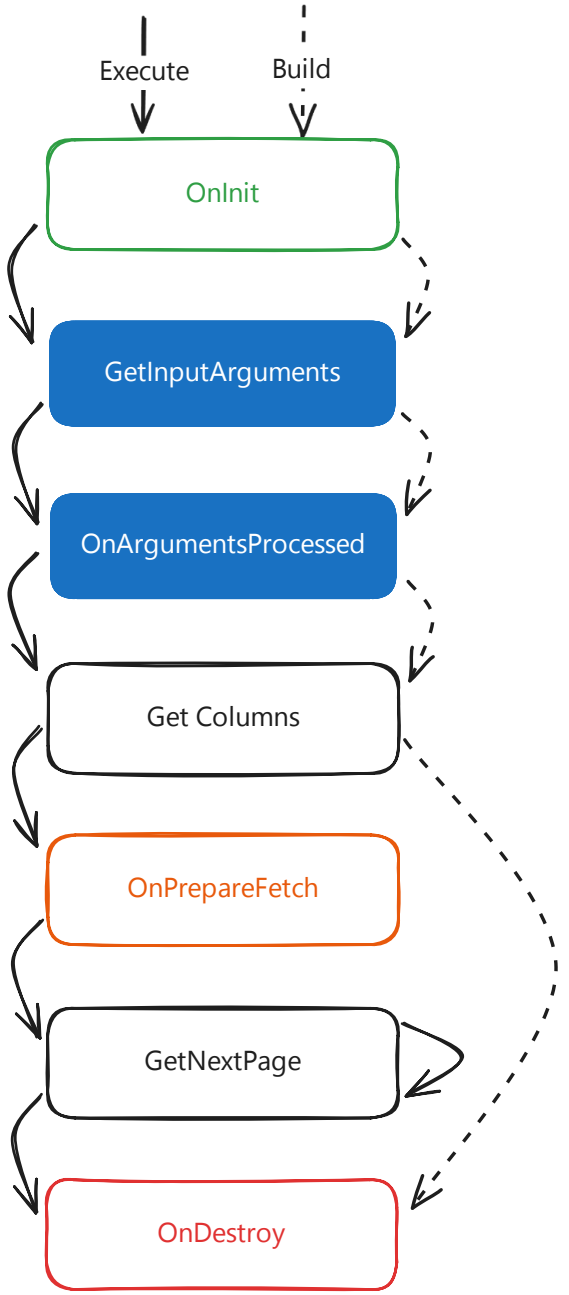


The life cycle

More stuff

- GetInputArguments
- OnArgumentsProcessed

- Provided from C# interface: IGQIInputArguments

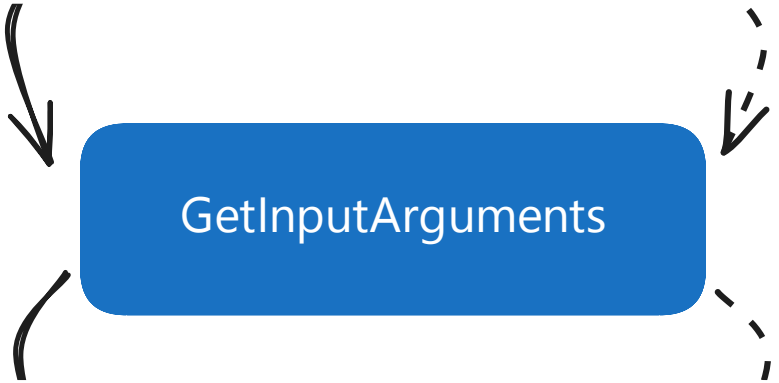


Define input arguments

GetInputArguments

- Possible arguments:
  - Text, numbers, dates, …
  - Lists, dropdowns

- Arguments can be required or optional



Retrieve argument values

OnArgumentsProcessed

- Check if arguments exist
- Argument validation
- Store argument values for later use

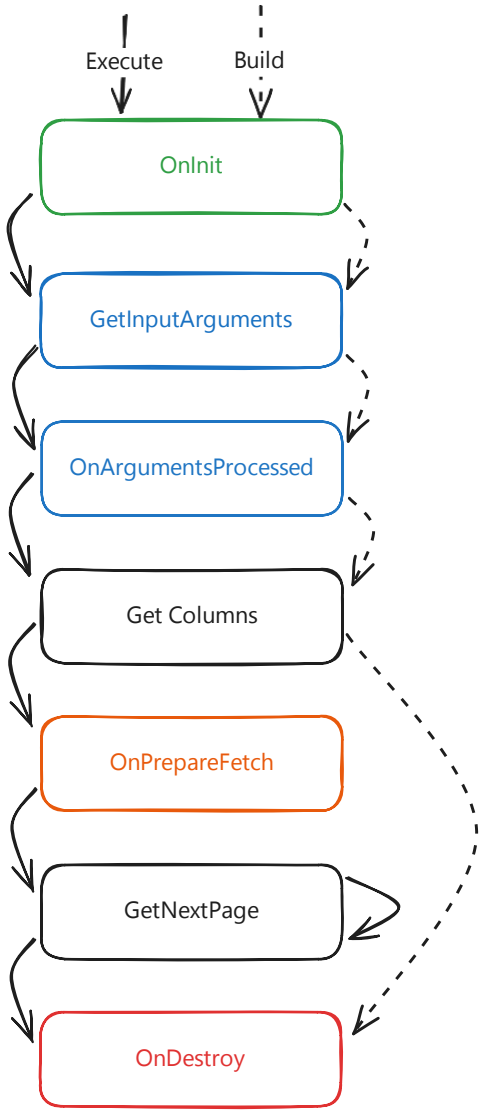
Let's add some arguments!

We can take in user input to modify our data source

Supporting real-time updates

Be open to change!

The life cycle

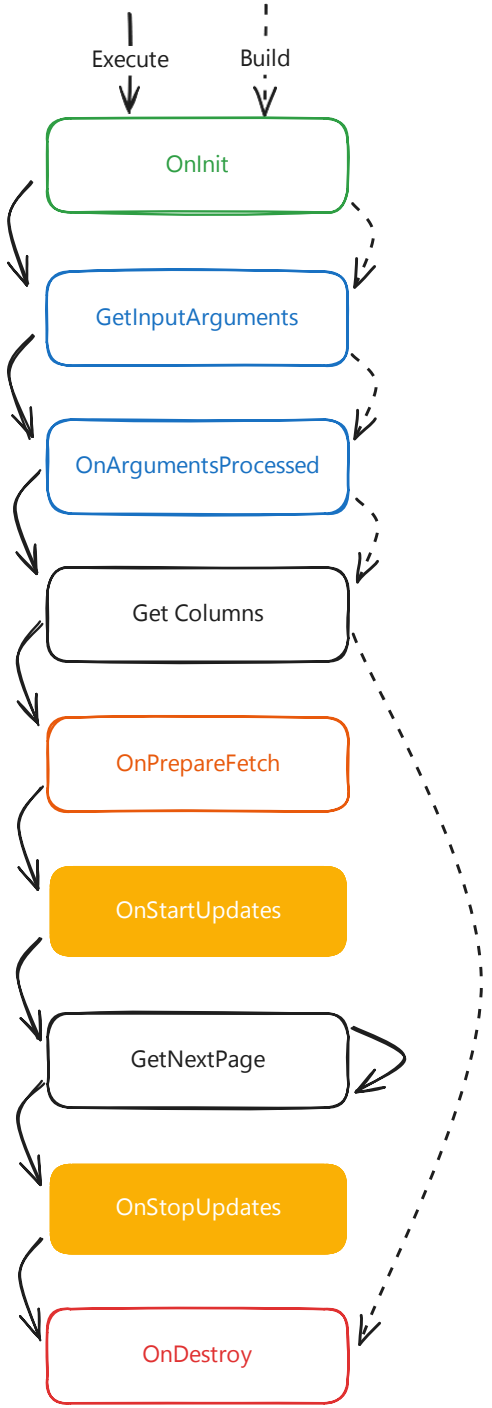


The life cycle

Updates!

- OnStartUpdates
- OnStopUpdates

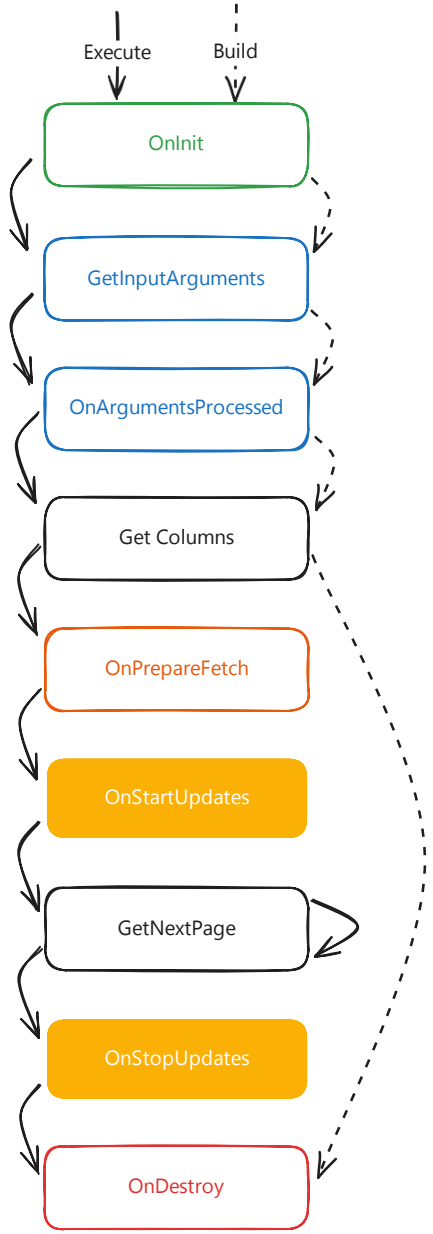
- Provided from C# interface: IGQIUpdateable



Prerequisites to support updates

Don't forget

- Enable updates on your component
- Give IDs to your rows to reference them later



Starting updates

OnStartUpdates

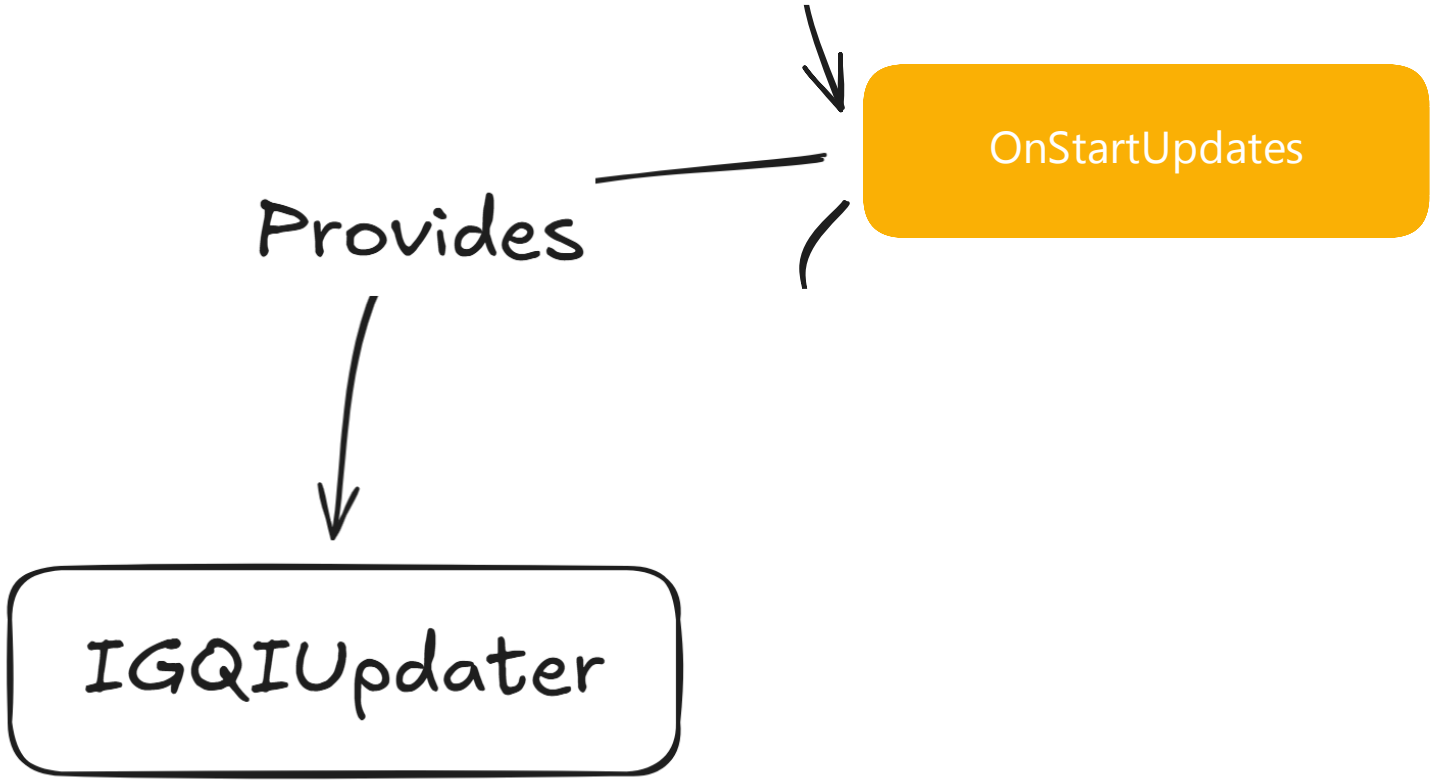
- Called once before the data is requested
- Provides an Updater object to publish updates

- Should be used to initialize anything needed to do updates

The updater

IGQIUpdater

- Update entire rows
- Update individual cells
- Remove existing rows
- Add new rows



Stopping updates

OnStopUpdates

- Called just before OnDestroy if OnStartUpdates was called

- Should be used to clean up anything that was initialized in OnStartUpdates

Let's update!

Connecting to DataMiner

Communicating with the DataMiner system

Everything connects

- OnInit provides a GQIDMS object
- Allows sending SLNet messages to request ANY DataMiner data
- Allows creating subscriptions
- https://www.nuget.org/packages/Skyline.DataMiner.Dev.Common/

Best practices

- Use the DIS template
- Consider caching when performing large computations (static memory or file or ...)
- Use paging when possible (lazy loading)
- Use real time updates instead of relying on refreshes from the client
- Define related data source in the same library

[ASK AWAY]

empower!