DEVELOPING A CUSTOM IMPORTOMATIC CONNECTOR: THE BASICS

IMPORT CONNECTOR

CONCEPT

At the core, an import connector is pretty basic; simply author a class that offers metadata for the data that will be retrieved, and wire in code that will "get the next record". ImportOmatic will do the leg work and simply repeatedly ask your connector to "get the next record" until your connector returns null; indicating it is done.

Of course most connectors will have a bit more complexity, like storing configuration settings (web service URL, API key/credentials, etc.), auto/pre-mapping fields, or performing custom Authorization logic, but at the root a simple .NET class library will get you plugged into the framework.

CREATE A CLASS LIBRARY

Create a VB/C# class library that contains at least one class that meets the following

- Is public
- Inherits from `OmaticIO.Base.Import.ImportSourceBase`
- Overrides at minimum:
    - Name
    - Author
    - Description
    - SourceID
    - Version
    - `InternalGetMetaData`
    - `InternalGetNextRecord`

TEST USING IMPORT SIMULATOR

Using import simulator you can confirm that the very basics of your connector are working. Place your compiled connector library in a subfolder of the simulator called IOMExtensions. Run the simulator and you should see your connector show in the Connectors dropdown.

Show Config button will call the configuration for your connector (if implemented).

The Run button will kick off a simulation that will fetch records from your connector until no more records are returned.

EXPORT CONNECTOR

CONCEPT

Export connector functionality for IOM is a little more complex than the import connector. One of the most important things to understand is that exporting requires 2 pieces, a source and one or more destinations. The **source will determine how and what data will come out of Raiser's Edge**. The **destination will then be responsible for making sure that data goes somewhere**.

For any export the user will need to decide the Source and the destination(s), but your connector does not necessarily have to provide both source and destination. Out of the box IOM supports a "SQL Source" which allows the user to write SQL to extract data from RE. We expect to offer an RE Query Source as well that will allow the user to select an RE Query as the source of data to export.

These built in Sources could allow your connector to simply provide an export destination (e.g. class that pushes data to a web service).

SOURCE

The export source will be responsible for identifying and retrieving what data needs to be exported.

CREATE A CLASS LIBRARY

Create a VB/C# class library that contains at least one class that meets the following

- Is public
- Inherits from `OmaticIO.Base.Export.ExportSourceBase`
- Overrides at minimum:
  - Name
  - Author
  - Description
  - SourceID
  - Version
  - `InternalGetMetaData`
  - `InternalGetNextRecord`

DESTINATION

The export destination is responsible for "doing something" with each record provided by the export source.

CREATE A CLASS LIBRARY

Create a VB/C# class library that contains at least one class that meets the following

- Is public
- Inherits from `OmaticIO.Base.Export.ExportDestinationBase`
- Overrides at minimum:
  - Name
  - Author
  - Description
  - DestinationID
  - Version
  - `InternalProcessRecord`

TEST USING EXPORT SIMULATOR

TODO: document the use of export simulator