Introduction

This document assumes a familiarity with technologies such as RDF, SWP, React and GraphQL.

Introduced in EDG 6.4 the Asset Collection Table has migrated away from its previous implementation of GQL and DataTables in favor of a React solution based on react-tables. This transition was done to align the user interface with the feature sets that were being developed for the new editors. With this transition the extension process for the table has been modified. This document will cover how to define a new column as part of the system schema and render it in the asset collection table.

There are three layers of technologies involved in render the asset collection table.

- SWP/HTML hook
    - The table itself is render on the client side via React, but SWP continues to deliver the HTML container where the table will eventually render. This hook is defined in `teamwork:ProjectTypeSectionTable`, this element is overridable and is what allows an extension to have complete control over the rendered UI. As the 'README' explains, this element is what needs to be overridden with standard SWP paradigms. ('ui:overrides')
- GraphQL
    - TopBraid EDG provides a system level GraphQL endpoint, in this document we will be adding a new field that can expand the existing schema and then once available make use of the new data in a column.
- React
    - Once the SWP/HTML hook has been delivered to the browser React takes over, fetching the data, building state, rendering the table.

Whats in the package

This document is meant to be paired with the `sample-asset-collection-table` project. This project is a valid TopBraid Composer and some files such as `sample-asset-collection-table.ui.ttlx` will be easier to follow from within TopBraid Composer.

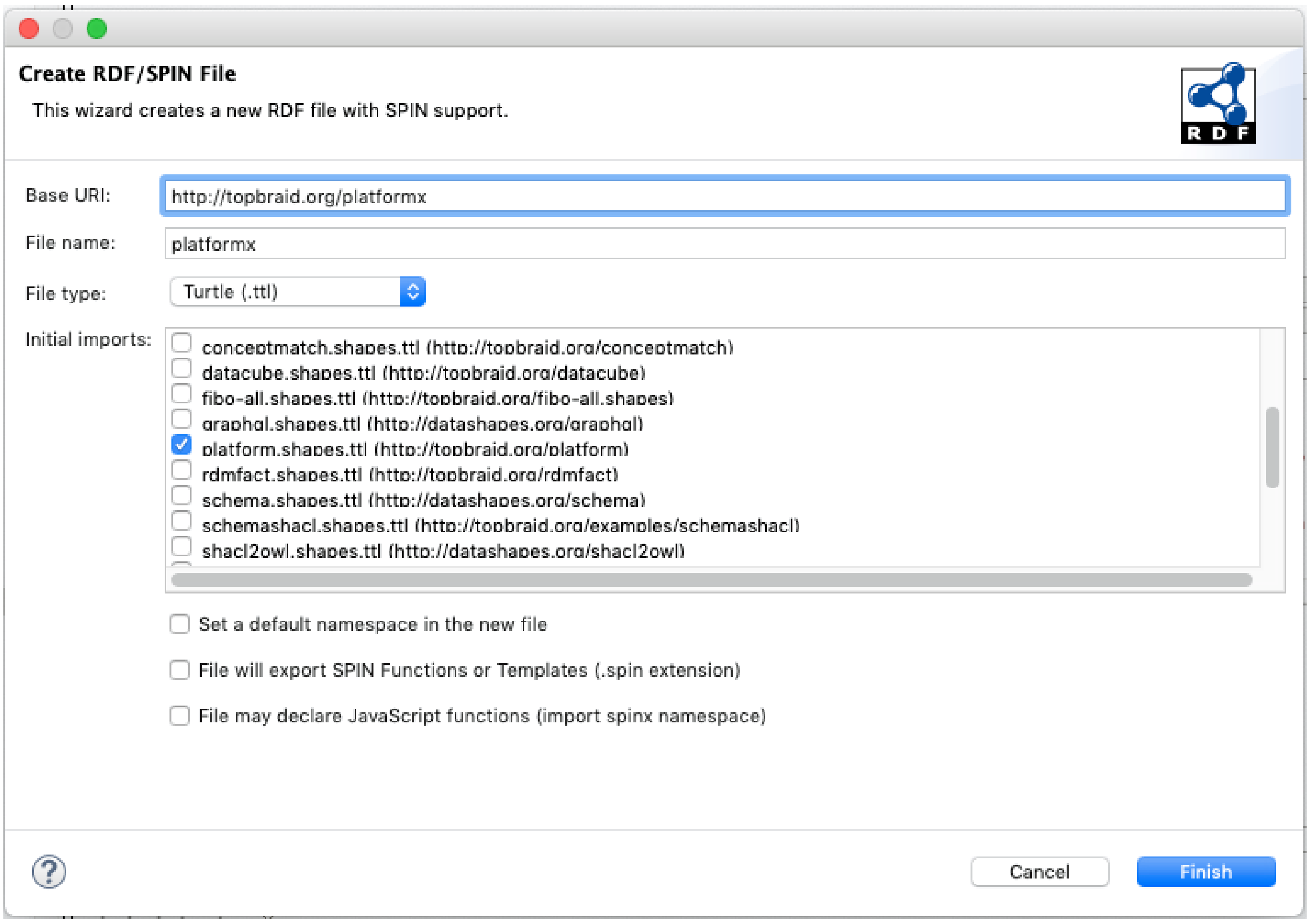
```
platformx.ttl
sample-asset-collection-table.ui.ttlx
- app.www
- assets
-- index.js
-- images
-- js
--- sampleAssetCollectionTable.jsx
```

Expanding the Schema

This section explains how to add a new field to the system level schema. Targeting the field 'official name'.

The system level schema that generates the _ GraphQL endpoint can be extended through a custom ontology. The file `platformx.ttl` is an example of this extension mechanism. Once deployed to Topbraid EDG the GraphQL engine will automatically include the additional statements. The steps below explain how the `platformx.ttl` ontology was defined. But the final result can always be seen through opening `platformx.ttl` in TopBraid Composer.

- Within a TBS extension project define a new RDF file, make sure to use the following baseURI `http://topbraid.org/platformx`. This is necessary for the GraphQL engine to find the extension. Note: this baseURI can be utilized exactly once for a given installation so all extensions will need to share a common ontology.
- Import `http:topbraid.org/platform` which is required to target the existing platform definitions and then `http://topbraid.org/metadata` which is utilized to target `metadata:officialName`.


Create RDF/SPIN File

This wizard creates a new RDF file with SPIN support.

Base URI: http://topbraid.org/platformx

File name: platformx

File type: Turtle (.ttl)

Initial imports:
- [ ] conceptmatch.shapes.ttl (http://topbraid.org/conceptmatch)
- [ ] datacube.shapes.ttl (http://topbraid.org/datacube)
- [ ] fibo-all.shapes.ttl (http://topbraid.org/fibo-all.shapes)
- [ ] graphql.shapes.ttl (http://datashapes.org/graphql)
- [✓] platform.shapes.ttl (http://topbraid.org/platform)
- [ ] rdmfact.shapes.ttl (http://topbraid.org/rdmfact)
- [ ] schema.shapes.ttl (http://datashapes.org/schema)
- [ ] schemashacl.shapes.ttl (http://topbraid.org/examples/schemashacl)
- [ ] shacl2owl.shapes.ttl (http://datashapes.org/shacl2owl)

- [ ] Set a default namespace in the new file
- [ ] File will export SPIN Functions or Templates (.spin extension)
- [ ] File may declare JavaScript functions (import spinx namespace)

- Find the class `platform:Graph` and define a new property shape with standard SHACL syntax the resulting definition should look like this.

```
<http://topbraid.org/platform#Graph>
  sh:property [
      sh:path platformx:officialName ;
      sh:datatype xsd:string ;
```

```
        sh:maxCount 1 ;
        sh:name "officialName" ;
        sh:values [
            sh:prefixes <http://topbraid.org/metadata> ;
            sh:select """SELECT DISTINCT ?officialName
                         WHERE {
                            GRAPH $this {
                               $this metadata:officialName ?officialName
                           }
                         }  """ ;
          ] ;
      ] ;
.
```

Expanding the Table

Now that the schema has been extended we want to take advantage of this new data field. There are an infinite number of extension points that could be added to a view but to avoid that complexity the best solution is to just override the entire view. While this does add some coding it allows the extension developer complete control of the interface. The steps below highlight how the extension was defined.

Note: The file `sample-asset-collection-table.ui.ttlx` serves two main purposes 1) it defines how the extension point registers JavaScript/React components. 2) it handles any other SWP extensions.

- Assuming TopBraid Composer is running and the sample project has been imported open the file `sample-asset-collections-table.ui.ttlx`.
    - Navigate to the class definition for `sample-asset-collection-table:ProjectTypeSectionTable`.
    - Note this lass definition is a clone of`teamwork:ProjectTypeSectionTable`
    - The only difference outside of the name is in the ui:prototype; the'id' attribute of the div tags was updated `ac-react-table > sample-ac-react-table`.
    - The ui:overrides property was set to `teamwork:ProjectTypeSectionTable`.

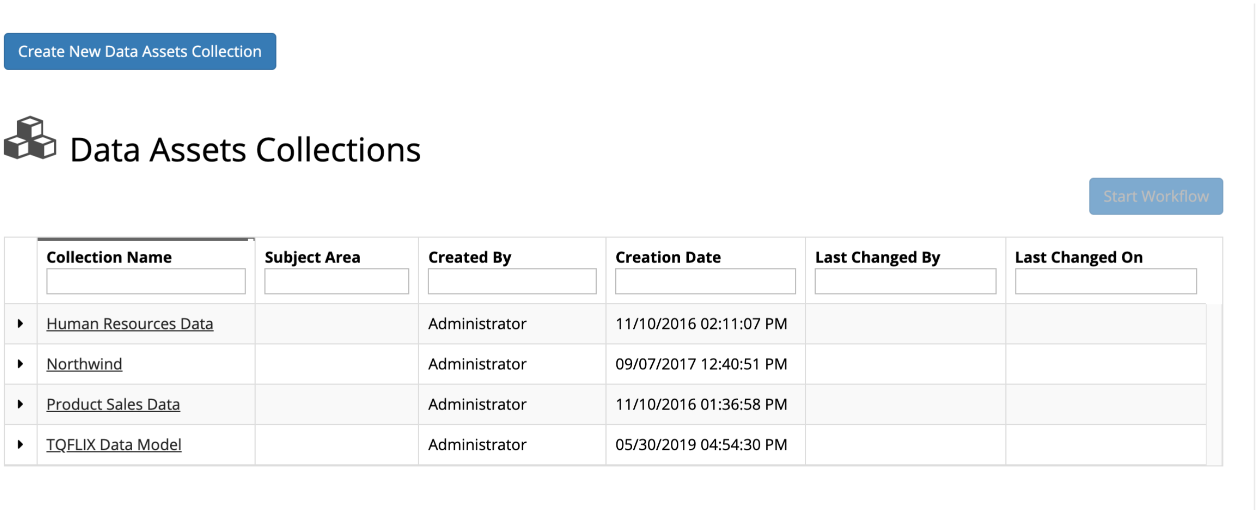
Note: These steps can occur in a standard editor (VSCode for example)

- Navigate to the file `samplesAssetCollectionTable.jsx` observe that this file is a clone of the core React component `AssetCollectionDataTableComponent` with some noted differences. 1) The imports have been adjusted to utilize the global namespaces 'TB', 'swa', 'graphql'. 2) The in making the clone the extension developer can pick and choose which features to utilize.
    - Find the function `SAMPLE_buildQueryTemplate` has been modified to remove some defaults but to include the new property `officialName`. This function is defining the GQL query that will be utilized to fetch the table information.
- Find the constant `SAMPLE_DEFAULT_COLUMNS`, this array contains the column definitions that will be utilized by react-table. As of the creation of this document TopBraid EDG makes use of react-

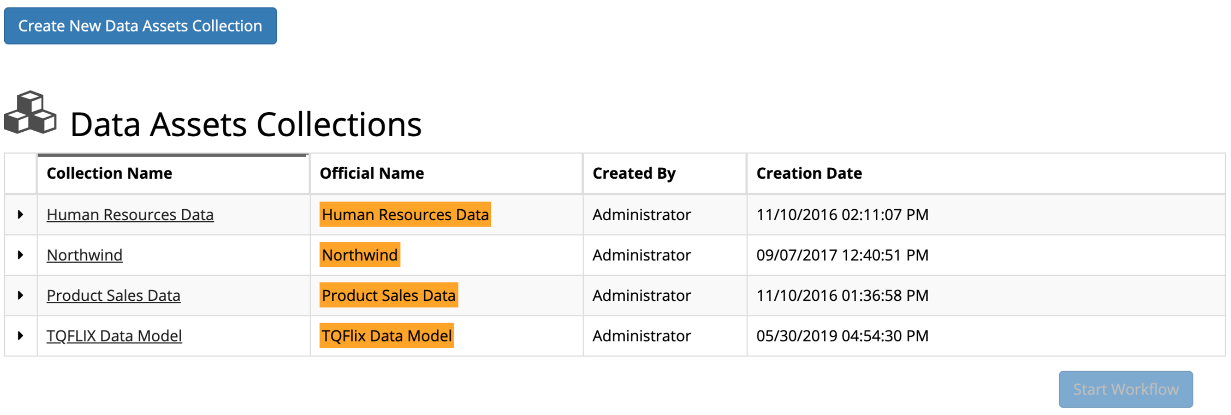
table version 7.0.3. Note the differences with the constant `DEFAULT_COLUMNS`, some definitions have been removed while one has been added for 'official name'.

- Observe that the `readystatechange` event handler, defined at the bottom of the file, has been updated to target the new div tags with id `sample-ac-react-table`.
- Build the extension
    - Because the extension is React based it will need transpiled and bundled prior to utilization. This sample provides the necessary configurations to build and bundle the extension.
    - From the command line:
        - `yarn install` - install's all of the necessary dependencies
        - `yarn run build` - will build the extension
        - `yarn run zip` - assuming the deployment target is an EDG instance that is running on tomcat you'll need this command to bundle the extension. If however you're running an EDG instance from TBC you can skip this step as the project will have access to the bundles.

The Result



	Collection Name	Subject Area	Created By	Creation Date	Last Changed By	Last Changed On
▸	Human Resources Data		Administrator	11/10/2016 02:11:07 PM		
▸	Northwind		Administrator	09/07/2017 12:40:51 PM		
▸	Product Sales Data		Administrator	11/10/2016 01:36:58 PM		
▸	TQFLIX Data Model		Administrator	05/30/2019 04:54:30 PM		



	Collection Name	Official Name	Created By	Creation Date
▸	Human Resources Data	Human Resources Data	Administrator	11/10/2016 02:11:07 PM
▸	Northwind	Northwind	Administrator	09/07/2017 12:40:51 PM
▸	Product Sales Data	Product Sales Data	Administrator	11/10/2016 01:36:58 PM
▸	TQFLIX Data Model	TQFlix Data Model	Administrator	05/30/2019 04:54:30 PM