Changed Record Queries

The Ed-Fi ODS / API platform contains data that gets updated frequently. The platform can track inserts, updates, and deletes, and surface those changes to client systems through a feature called changed record queries, or "change queries." Change queries allow client systems to narrow requests for data to only data that has changed since a specified point in time. This allows client systems to stay in sync with the ODS / API without having to pull a complete data set.

Client system interaction is documented in the Using the Changed Record Queries section of the API Client Developers' Guide.

Change queries is an optional feature and is turned off by default — but can be turned on through configuration. This documentation covers the essentials for platform hosts to enable and manage the feature.

Enabling Change Queries

Enabling the change query feature is easy and can be done through configuration. Note that the change requires a redeployment of the database (but a rebuild of the solution code is not required).

The feature is enabled on the deployed code by changing the Web.config file of the EdFi.Ods.WebApi project. This is the Web.config of the "Api" component of the deployed solution. The app setting "changeQueries:featureIsEnabled" should be set to the value "true". If the app setting doesn't exist, it should be created.

The following snippet shows the app setting:

```
<add key="changeQueries:featureIsEnabled" value="true" />
```

> When working in a development environment, you will need to change the value in the *Web.base.config* file, instead of *Web.config*, as the Web.config file is generated from the Web.config.base during the build process.
>
> If installing using the `EdFi.RestApi.Databasess.EFA` NuGet package, instead of deploying from the source code repository, then it is the `Databases.config` file that needs to be modified.

To enable change queries, the database must be also be updated using the provided scripts. These scripts will set up a new database schema and all the changes necessary to support change queries. All scripts supporting change queries will exist under a subfolder named "changes" inside the "Ods" target database folder (i.e., \Ed-Fi-ODS\Artifacts\MsSql\Structure\Ods\changes). Scripts will be generated for this feature by MetaEd for Ed-Fi Extension projects as well. Note that in development environments, initdev process automatically deploys the change query schema based on the "changeQueries:featureIsEnabled" flag in the Web.config file of the EdFi.Ods.WebApi project. This means that a development instance will typically not require any additional database scripts to be run.

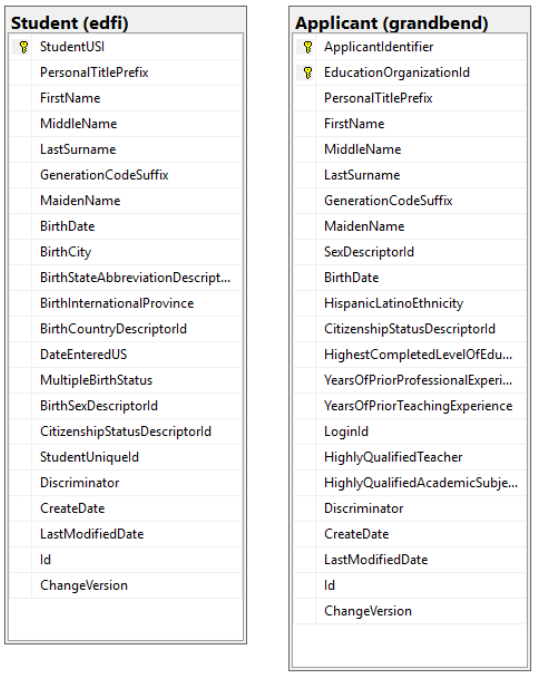
> Once the database has been updated to support change queries, there is no automated mechanism in place to remove this support or undo the changes.

In non-development environments, this feature must be enabled in the database by deploying the necessary change scripts to the target ODS database(s). The exact steps depend on your deployment method:

- If you are using the built-in deployment PowerShell scripts, update the Databases.config to set the "changeQueries:featureIsEnabled" flag to "true". Performing a deployment on top of an existing ODS database after setting this flag will correctly execute the change queries scripts, which perform the necessary updates to enable change queries functionality.
- If the built-in deployment scripts are not being used, all scripts under the "changes" inside the "Ods" target database folder (Ed-Fi-ODS-Implementation\Application\EdFi.Ods.Standard\Artifacts\MsSql\Structure\Ods\Changes) must be run against the ODS, including the extension version of the scripts (Ed-Fi-ODS-Implementation\Application\{your extension project}\Artifacts\MsSql\Structure\Ods\changes).

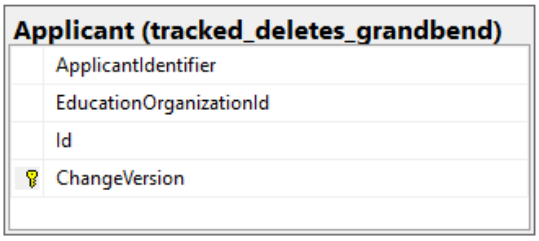
Technical Details

The change queries function uses basic versioning concepts. A global version counter is introduced using a Sequence object, and each table representing the top-level entities of a data domain are given a ChangeVersion column to represent the current version of the entity instance. These columns are set up with triggers to ensure they are automatically updated based on the latest value of the sequence on all inserts and updates, whether they come from the API or through scripts. Queries done against the API to find changed records function by adding an additional where clause to based on this ChangeVersion column.



Supporting delete tracking requires another common change tracking concept, using a "tombstone" table. This concept involves tracking all deletes and storing a record of what was deleted, to allow future querying against deleted records. In this case, the pattern is implemented using a delete trigger on the main table, and the resource id and primary key of the deleted record are then stored in the tracking table for deletes. API requests to retrieve information on deletes queries against these tables to retrieve the resource ids of deleted entity instance.

Note that there is a concern in use cases with very high volume of deletes or for very long running ODS instances where the size of these delete tracking tables can get very large, taking up a disproportionate percentage of the database storage. In that case, these tables can be truncated periodically, the only risk being losing visibility into any deletes that are removed by this truncation. Given the typical uses cases of using an ODS per school year, no automatic truncation was introduced by default since it's unlikely to be an issue for most implementations.



The patterns described above are used for both the standard, as-shipped Ed-Fi ODS database tables, as well as tables generated to serve Ed-Fi Extensions. By using METAED (the recommended method for extending the Ed-Fi ODS / API), scripts are generated to support the ChangeVersion column, insert /update/delete triggers, and delete tracking tables. This allows hosts to have consistent support for the feature across their entire API, even for new top-level entities introduced by Extension projects.