Extending the ODS / API Data Model

The basic pattern for adding extensions to the Ed-Fi ODS / API data model is to develop and place customizations into the Ed-Fi-ODS-Implementation directory. The top levels of this directory mirror the standard functionality that is contained in the Ed-Fi-ODS directories. When developers complete the development for an extension to the ODS / API data model, re-running the initialize development environment steps (discussed in the Getting Started documentation) will use code generation to automatically add your data extensions to the Ed-Fi ODS / API.

Implementing an extension involves making changes to the following components:

- XSD Schema
- Database Scripts
- API Metadata

Implementation details for each of these components are described below. In addition, see the How To: Extend the ODS / API - Student Transportation Example and How To: Extend the Ed-Fi ODS / API - Student Transcript Example articles for a complete walkthrough.

XSD Schema

The Ed-Fi Data Standard provides XSD interchange schema for a number of data exchange scenarios. ODS / API Implementers can extend these schema to customize the data transfer for their particular needs. See the Ed-Fi Data Standard Developers' Guide for information on how to extend the standard. Extension schema files go in the \Ed-Fi-ODS-Implementation\Extensions\Schemas directory.

The Ed-Fi Standard Interchange Schema files required for the ODS / API system are downloaded via a NuGet feed. At build time, the schema files from the standard are copied into a working directory (Ed-Fi-ODS\Application\schema.codegen), then any extension schema files are copied on top of the files of the same directory. Extensions to the Ed-Fi schema are made in the EXTENSION-Ed-Fi-Core.xsd file, which references Ed-Fi-Core.xsd to reference the new extensions. Extension XSD file names must start with EXTENSION and all complex types in the XSD must start with EXTENSION-.

Database Scripts

The ODS database scripts are located in their respective \Database\Structure\EdFi and \Database\Data\EdFi directories. The Ed-Fi-ODS repository contains the standard (i.e., un-customized) data structures, while the Ed-Fi-ODS-Implementation repository contains extensions. During the initialize development environment process, a set of standard scripts are run against the database, followed by the implementation scripts. The structure scripts create the database objects, and the database tables are populated by the data scripts. For example, Ed-Fi type tables are populated during this process from scripts in the Ed-Fi-ODS\Database\Data\EdFi directory.

Extension schema files need to be named in this pattern: 0001-description.sql, where 0001 is incremented for each additional sql file. Each script is run in numerical order; scripts that have been previously run in a given database are skipped (by number).

You should put CREATE or ALTER scripts in the database implementation directories that reflect the extensions to the XSD Schema. If you need to provide additional database objects to support your customization, they should be placed here as well. You may also populate descriptors and education organizations using this technique.

Populated Sample Data

Your database extension scripts are automatically run against the sample database (minimal or populated databases). The populated sample database is retrieved from a NuGet package. However, this database contains education organizations and descriptors that you may not wish to use. If you wish to provide your own sample data, the EdFi.Samples.Ods.* NuGet should not be allowed to populate the “EdFi_Ods_Populated_Template” database during the initialize development environment process.

API Metadata

This section describes several metadata files used by the code generation to generate the Ed-Fi ODS / API, some of which may require modification when extending the ODS / API.

DomainMetadata.xml

DomainMetadata.xml is an API metadata file used to group the ODS tables of Domain Entities, Associations, Descriptors, and Enumerations into API-level aggregates. These aggregates map directly to resources exposed by the REST API.

Developers' Guide Contents

Find out more about how to develop platforms based on the Ed-Fi ODS / API v2.4:

The file contains a collection of `<Aggregate>` tags, each one defining a single aggregate. Inside each `<Aggregate>` tag is a list of `<Entity>` tags that specify the ODS tables associated with the aggregate.

SkipReferenceMetadata.xml

SkipReferenceMetadata.xml is an API metadata file used to indicate identity reference paths that the API should ignore when there are multiple paths between two entities. The API needs this information to define a well-organized tree structure without unnecessarily deep data hierarchies or circular references. It is expected that the need for an extension to modify SkipReferenceMetadata.xml will be rare.

The file contains a collection of `<SkipReferenceType>` tags, each one defining a single identity reference path to ignore. Inside each `<SkipReference>` tag is an ordered list of `<NormalizedSchemaTypeName>` tags. These tags declare the path to ignore by naming the XSD elements in the path. Note that the elements are always declared in reverse order.

PredefinedContextMetadata.xml

PredefinedContextMetadata.xml is an API metadata file used to override the standard pattern for naming columns in the ODS. It is also used to specify where key unification is related to the naming override. It is expected that the need for an extension to modify PredefinedContextMetadata.xml will be rare.

The file contains a collection of `<ContextMetadata>` tags, where each one defines a single naming override. Inside each `<ContextMetadata>` is a `<ParentElementTag>` that specifies the XSD element representing the parent entity, an `<ElementName>` tag that specifies the XSD element representing the field on the parent entity, and a `<Context>` tag that specifies the naming override. The `<ContextMetadata>` tag can optionally include a `<UnifiedElementName>` tag that specifies the XSD identity element to which this rename is related for key unification.

InterchangeOrderMetadata.xml

InterchangeOrderMetadata.xml is an API metadata file used to indicate data load order dependencies for Domain Entities, Associations, and Descriptors, grouped by interchange. InterchangeOrderMetadata.xml is almost always overridden in an implementation because decisions about security configurations can impact proper load order. An extension InterchangeOrderMetadata.xml replaces the core version. It is not additive.

The file contains an ordered list of `<Interchange>` tags that specify the load order for each Ed-Fi interchange. Inside each `<Interchange>` tag is an ordered list of `<Element>` tags that specify the load order for each Domain Entity, Association, and Descriptor in the interchange. Not all interchange elements need to be declared. Omitted interchange elements are implicitly specified as being loaded after the declared elements, but in an undefined order.

EdOrgReferenceMetadata.xml

EdOrgReferenceMetadata.xml is an API metadata file used to indicate which primary key on an EducationOrganization subclass table in the ODS maps to EducationOrganizationId. It is expected that the need for an extension to modify EdOrgReferenceMetadata.xml will be rare.

The file contains a list of `<EdOrgReference>` tags, each of which refers to an EducationOrganization subclass table. Inside each `<EdOrgReference>` tag is a `<type>` tag that contains the name of the subclass table, and a `<key>` tag that contains the name of the column that maps to EducationOrganization.

PredefinedNoForeignKeyMetadata.xml

PredefinedNoForeignKeyMetadata.xml is an API metadata file used to override the standard pattern for foreign keys in the ODS. The file specifies where foreign key relationships between tables should be suppressed. It is expected that the need for an extension to modify PredefinedNoForeignKeyMetadata.xml will be rare.

The file contains a collection of `<NoForeignKeyMetadata>` tags, each of which defines a single foreign key suppression. Inside each `<NoForeignKeyMetadata>` is a `<ParentElementName>` that specifies the XSD element representing the parent entity, and an `<ElementName>` tag that specifies the XSD element representing the field on the parent entity that is the reference whose foreign key on the ODS should be suppressed.

Naming Conventions for Extensions

Care should be taken with the use of acronyms in extensions.

- As general guidance, it is recommended that implementations use the Pascal or CamelCase naming convention for acronyms over two characters in length. For more information, see .NET Framework Design Guidelines.

- There is a known bug related to inconsistent handling of acronyms in Swagger metadata (described here). Since addressing the bug would affect JSON payloads, a fix has been deferred to the Ed-Fi ODS / API v3.0 release. In the meantime, it is recommended that all-caps acronyms are not used as the leading characters in an extension (e.g., don't name an extension "IEPxxxx", rather use "Iepxxxx" instead).