How To: Configure Dashboard Operational Context

Technical Overview

Operational Contexts allow for mapping between sets of Ed-Fi Descriptors. In practice, the Ed-Fi ETL internally runs calculations based on Ed-Fi-provided Descriptors. Therefore, if you have custom Descriptors in use in your source ODS, you'll need to map these custom Descriptors to the as-shipped Ed-Fi Descriptors in order for your data to be included in calculations.

The process outlined below shows how to create an operational context that will be recognized by the ETL process, and map custom Descriptors through this operational context.

> **Fully Custom Descriptors**
>
> Fully custom descriptors (i.e., wholly new descriptors created by extension to the ODS) are not supported in the ETL. You do not need to map these Descriptors as they have no equivalents in the as-shipped Ed-Fi technology.

The high-level steps are:

Details follow.

Step 1. Create the Operational Context

Note this script must be run as-is. The values provided below are used by the ETL code.

```
INSERT INTO interop.OperationalContext (OperationalContextUri, DisplayName)
VALUES ('uri://ed-fi.org/Dashboard', 'Ed-Fi Dashboard Operational Context')
```

Step 2. Identify Descriptors that May Need Mapping

The below query will show you any Descriptors defined in a non-Ed-Fi namespace that are not currently mapped in the Dashboard Operational Context. These are all candidate Descriptors you should consider mapping. Note that not all Descriptors need to be mapped as not all Descriptors are used by the ETL process.

If the result of this query is empty, you have no more custom Descriptors that are unmapped in the Dashboard Operational Context.

```
/*
Retrieve ALL of the unmapped custom descriptors
*/
SET NOCOUNT ON

DECLARE @AllUnmappedCustomDescriptors TABLE
(
    [CodeValue] NVARCHAR(50) NOT NULL,
    [DescriptorTable] NVARCHAR(255) NOT NULL,
    UNIQUE ([CodeValue], [DescriptorTable])
)

DECLARE @descriptor_table NVARCHAR(255)

DECLARE descriptor_tables CURSOR FOR
SELECT t.[name]
FROM [sys].[tables] t INNER JOIN [sys].[schemas] s
ON t.schema_id = s.schema_id
WHERE t.[name] LIKE '_%Descriptor'
AND s.[name] = 'edfi'
ORDER BY t.[name]

OPEN descriptor_tables

FETCH NEXT FROM descriptor_tables
INTO @descriptor_table

WHILE @@FETCH_STATUS = 0
BEGIN
    INSERT INTO @AllUnmappedCustomDescriptors (DescriptorTable, CodeValue)
    EXEC('SELECT '''+@descriptor_table+''' AS DescriptorTable, CodeValue
    FROM edfi.Descriptor
    WHERE Namespace LIKE ''%'+@descriptor_table+'''
    AND CodeValue IN (
        SELECT CodeValue
        FROM edfi.'+@descriptor_table+' dt
        INNER JOIN edfi.Descriptor d
        ON dt.'+@descriptor_table+'Id = d.DescriptorId
        WHERE Namespace NOT LIKE ''uri://ed-fi.org/%'')
    GROUP BY CodeValue
    HAVING COUNT(*) = 1')

    FETCH NEXT FROM descriptor_tables
    INTO @descriptor_table

END
CLOSE descriptor_tables;
DEALLOCATE descriptor_tables;

SELECT d.DescriptorId, d.Namespace, m.DescriptorTable, m.CodeValue
FROM @AllUnmappedCustomDescriptors m
INNER JOIN edfi.Descriptor d
ON d.Namespace LIKE '%/'+DescriptorTable
AND d.CodeValue = m.CodeValue
WHERE NOT EXISTS (
    SELECT *
    FROM interop.OperationalContextSupport s
    WHERE TargetOperationalContextUri = 'uri://ed-fi.org/Dashboard'
    AND SourceDescriptorUri = d.[Namespace] + '#' + d.CodeValue
)
ORDER BY DescriptorTable

SET NOCOUNT OFF
```

Step 3. Add Descriptor Mapping to the Operational Context

Modify the `@descriptor_table` and `@edfi_code_value` variables below, and replace the `SELECT 'Math'` line with the custom CodeValue you want mapped to `@edfi_code_value`. If you have more than one custom CodeValue to map to the same `@edfi_code_value`, uncomment and use the `UNION ALL` syntax on the subsequent lines.

```
DECLARE @DashboardOperationalContextNamespace NVARCHAR(255)
SET @DashboardOperationalContextNamespace = 'uri://ed-fi.org/Dashboard'

DECLARE @descriptor_table NVARCHAR(255)
DECLARE @edfi_code_value NVARCHAR(255)

IF EXISTS (SELECT * FROM tempdb.sys.objects WHERE [name] =
'##custom_code_values' AND [type] = 'U')
    DROP TABLE ##custom_code_values

CREATE TABLE ##custom_code_values([CodeValue] NVARCHAR(50) NOT NULL)

DECLARE @edfi_descriptor_id INT

/*
Change the values below depending on which descriptors you want to map.
*/

SET @descriptor_table = 'AcademicSubjectDescriptor' -- The Descriptor Type
SET @edfi_code_value = 'Mathematics'                -- The Ed-Fi CodeValue
that will be mapped TO

INSERT INTO ##custom_code_values (CodeValue)
SELECT 'Math'                                       -- The Custom
Descriptor CodeValue you want mapped to @edfi_code_value defined above
--UNION ALL                                          -- (Optional) Add
'UNION ALL' if there are multiple CodeValues to map
--SELECT 'Matematicas'                               -- (Optional) Any
Additional Custom CodeValues that will map to @edfi_code_value (Add 'UNION
ALL' if there are more)

DECLARE @descriptor_query NVARCHAR(MAX)
SET @descriptor_query = 'SELECT @edfi_descriptor_id = (SELECT TOP 1
DescriptorId FROM edfi.Descriptor d
    INNER JOIN edfi.'+ @descriptor_table + ' t
    ON d.descriptorId = t.'+ @descriptor_table +'Id
    WHERE d.CodeValue ='''+@edfi_code_value+'''
    AND d.Namespace LIKE ''uri://ed-fi.org/%'')'

EXEC sp_executesql @descriptor_query, N'@edfi_descriptor_id INT OUT',
@edfi_descriptor_id out

IF EXISTS(SELECT * FROM interop.OperationalContextDescriptorUsage WHERE
OperationalContextUri = @DashboardOperationalContextNamespace AND
DescriptorId = @edfi_descriptor_id)
BEGIN
    DECLARE @error_message NVARCHAR(MAX)
    SET @error_message = 'Mapping already exists for the Ed-Fi Descriptor
' + @descriptor_table + ' with Code Value ' + @edfi_code_value
    RAISERROR(@error_message , 20, 1) WITH LOG;
END

/*
Create Descriptor Mapping using the values provided above using either
Equivalence Groups or Generalization
*/
DECLARE @DescriptorsToMap TABLE
(
    [DescriptorId] INT NOT NULL
)

INSERT INTO @DescriptorsToMap (DescriptorId)
EXEC('SELECT DISTINCT DescriptorId FROM edfi.Descriptor d
    INNER JOIN edfi.'+ @descriptor_table + ' t
    ON d.DescriptorId = t.'+ @descriptor_table +'Id
```

```
    INNER JOIN ##custom_code_values c
    ON d.CodeValue = c.CodeValue')

/* Check if mapping already exists for a custom descriptor */
IF EXISTS (SELECT * FROM interop.DescriptorEquivalenceGroupAssignment a
           INNER JOIN @DescriptorsToMap d
           ON a.DescriptorId = d.DescriptorId)
BEGIN
    RAISERROR('Descriptor mapping already exists for some or all of the
custom descriptors provided.', 20, 1) WITH LOG;
END

BEGIN TRANSACTION;

BEGIN TRY
    INSERT INTO @DescriptorsToMap (DescriptorId) VALUES
(@edfi_descriptor_id)

    INSERT INTO interop.OperationalContextDescriptorUsage
(OperationalContextUri, DescriptorId)
    VALUES (@DashboardOperationalContextNamespace, @edfi_descriptor_id)


    DECLARE @use_generalization_mapping BIT
    SET @use_generalization_mapping = (SELECT CASE WHEN COUNT(*) > 1 THEN
1 ELSE 0 END
                                       FROM ##custom_code_values)

    IF(@use_generalization_mapping=1)
    BEGIN
        DECLARE @EdFiDescriptorGeneralizationGroupMembers TABLE
        (
            [DescriptorId] INT NOT NULL,
            [GeneralizationDescriptorEquivalenceGroupId] UNIQUEIDENTIFIER
NOT NULL,
            [DescriptorEquivalenceGroupId] UNIQUEIDENTIFIER NOT NULL
        )

        DECLARE @GeneralizationDescriptorEquivalenceGroupId
UNIQUEIDENTIFIER
        SET @GeneralizationDescriptorEquivalenceGroupId = NEWID()

        INSERT INTO @EdFiDescriptorGeneralizationGroupMembers
(DescriptorId, GeneralizationDescriptorEquivalenceGroupId,
DescriptorEquivalenceGroupId)
        SELECT DescriptorId, @GeneralizationDescriptorEquivalenceGroupId,
NEWID()
        FROM @DescriptorsToMap

        INSERT INTO [interop].[DescriptorEquivalenceGroup]
(DescriptorEquivalenceGroupId)
        VALUES (@GeneralizationDescriptorEquivalenceGroupId)

        MERGE [interop].[DescriptorEquivalenceGroup] AS [TARGET]
        USING
            (
                SELECT [DescriptorEquivalenceGroupId]
                FROM @EdFiDescriptorGeneralizationGroupMembers
            ) AS [SOURCE]
        ON ([TARGET].[DescriptorEquivalenceGroupId] = [SOURCE].
[DescriptorEquivalenceGroupId])
        WHEN NOT MATCHED BY TARGET THEN
            INSERT ([DescriptorEquivalenceGroupId], [CreateDate],
[LastModifiedDate], [Id])
            VALUES
            (
                [SOURCE].[DescriptorEquivalenceGroupId], GETDATE(),
GETDATE(), NEWID()
            );
```

```
        MERGE [interop].[DescriptorEquivalenceGroupGeneralization] AS
[TARGET]
        USING
            (
                SELECT [GeneralizationDescriptorEquivalenceGroupId],
                       [DescriptorEquivalenceGroupId]
                FROM @EdFiDescriptorGeneralizationGroupMembers
            ) AS [SOURCE]
        ON ([TARGET].[DescriptorEquivalenceGroupId] = [SOURCE].
[DescriptorEquivalenceGroupId])
        WHEN NOT MATCHED BY TARGET THEN
            INSERT
                (
                    [GeneralizationDescriptorEquivalenceGroupId],
                    [DescriptorEquivalenceGroupId],
                    [CreateDate],
                    [LastModifiedDate],
                    [Id]
                )
            VALUES
            (
                [SOURCE].[GeneralizationDescriptorEquivalenceGroupId],
[SOURCE].[DescriptorEquivalenceGroupId], GETDATE(), GETDATE(), NEWID()
            );

        MERGE [interop].[DescriptorEquivalenceGroupAssignment] AS [TARGET]
        USING
            (
                SELECT [DescriptorId], [DescriptorEquivalenceGroupId]
                FROM @EdFiDescriptorGeneralizationGroupMembers
            ) AS [SOURCE]
        ON ([TARGET].[DescriptorEquivalenceGroupId] = [SOURCE].
[DescriptorEquivalenceGroupId]
            AND [TARGET].[DescriptorId] = [SOURCE].[DescriptorId])
        WHEN NOT MATCHED BY TARGET THEN
            INSERT ([DescriptorId], [DescriptorEquivalenceGroupId],
[CreateDate], [LastModifiedDate], [Id])
            VALUES
            (
                [SOURCE].[DescriptorId], [SOURCE].
[DescriptorEquivalenceGroupId], GETDATE(), GETDATE(), NEWID()
            );
    END
    ELSE
    BEGIN
        DECLARE @EdFiDescriptorEquivalenceGroupMembers TABLE
        (
            [DescriptorId] INT,
            [DescriptorEquivalenceGroupId] UNIQUEIDENTIFIER NOT NULL
        )

        DECLARE @DescriptorEquivalenceGroupId UNIQUEIDENTIFIER
        SET @DescriptorEquivalenceGroupId = NEWID()

        INSERT INTO @EdFiDescriptorEquivalenceGroupMembers (DescriptorId,
DescriptorEquivalenceGroupId)
        SELECT DescriptorId, @DescriptorEquivalenceGroupId
        FROM @DescriptorsToMap

        MERGE [interop].[DescriptorEquivalenceGroup] AS [TARGET]
        USING
            (
                SELECT DISTINCT [DescriptorEquivalenceGroupId]
                FROM @EdFiDescriptorEquivalenceGroupMembers
            ) AS [SOURCE]
        ON ([TARGET].[DescriptorEquivalenceGroupId] = [SOURCE].
[DescriptorEquivalenceGroupId])
        WHEN NOT MATCHED BY TARGET THEN
            INSERT ([DescriptorEquivalenceGroupId], [CreateDate],
[LastModifiedDate], [Id])
            VALUES
```

```
            (
                [SOURCE].[DescriptorEquivalenceGroupId], GETDATE(),
GETDATE(), NEWID()
            );

        MERGE [interop].[DescriptorEquivalenceGroupAssignment] AS [TARGET]
        USING
            (
                SELECT [DescriptorId], [DescriptorEquivalenceGroupId]
                FROM @EdFiDescriptorEquivalenceGroupMembers
            ) AS [SOURCE]
        ON ([TARGET].[DescriptorEquivalenceGroupId] = [SOURCE].
[DescriptorEquivalenceGroupId]
            AND [TARGET].[DescriptorId] = [SOURCE].[DescriptorId])
        WHEN NOT MATCHED BY TARGET THEN
            INSERT ([DescriptorId], [DescriptorEquivalenceGroupId],
[CreateDate], [LastModifiedDate], [Id])
            VALUES
            (
                [SOURCE].[DescriptorId], [SOURCE].
[DescriptorEquivalenceGroupId], GETDATE(), GETDATE(), NEWID()
            );
    END
END TRY

BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;
END CATCH;

IF @@TRANCOUNT > 0
    COMMIT TRANSACTION;

DROP TABLE ##custom_code_values
```

Step 4. Repeat Until All Descriptors Are Mapped

Go to Step 2 and repeat for any remaining values you want mapped.