Using Mapping Data Flows with CDM in ADF and Synapse

Private Preview

5/5/2020

Overview

Mapping data flows, available in ADF and Synapse, can be used to read from and write to entities in a CDM Folder. CDM entities can be used as data sources and sinks in data flows like any other supported sources and sinks, allowing data to be transformed to and from entities using the full set of mapping and transformation capabilities supported by mapping data flows.

CDM support and the GitHub linked service are currently in Private Preview, for limited customer evaluation. **During private preview, use of this feature in production applications is not recommended or supported.** Details of the feature and the user experience will likely change before this is released.

Mapping data flows uses a 'known schema to known schema' approach. The entity definitions referenced must have been pre-defined out of band. See CDM documentation for help in defining CDM documents using CDM 0.9 or greater. https://docs.microsoft.com/en-us/common-data-model/

Enabling CDM support

To activate the preview, add the following switch to the ADF or Synapse editor URL: **&feature.cdm=true**

Capabilities

The following capabilities are supported in the private preview release.

- Supports CDM folders in ADLS gen2.
- Supports **CSV data files**; Parquet will be supported later.
- Supports **reading from manifest and model.json** and **writing to manifest files**.

Supported scenarios

The following scenarios are supported:

- Reading data from entities in CDM folders for mapping and transforming
- Writing data to entities in a CDM folder using entity definitions from a published model file

Data partition locations can be configured when writing to a CDM entity.

How to reference a CDM entity in a mapping data flow and execute

1. Create a pipeline with a mapping data flow.
2. In the mapping data flow, add a data source to read from an entity in a CDM folder and/or a data sink to write to an entity in a CDM folder.
3. Configure the data source or data sink to connect to the entity in the CDM folder (see below).
4. Connect the data source and sink in the mapping data flow like any other source or sink to map and transform data to or from CDM.

5. Execute the pipeline to test (use a debug environment to speed up testing).

More information is provided below.

Configuring a CDM data source or data sink

Entity data sources and sinks are defined by **configuring use of a Linked Service** associated with the ADLS gen2 storage account or a GitHub account containing the CDM folder. You **do not create and use a Dataset** for a CDM entity.

Working with Manifest files (read and write)

ADF mapping data flows support read and write to CDM folders described by a manifest file (*.manifest.cdm.json).

Referencing the CDM entity definition

Both the data source and sink configuration require a reference to the entity in a CDM definition file and to the manifest in the CDM folder. The referenced entity definition can be independent of the CDM folder and manifest. For example, the entity definition might part of an enterprise CDM model managed centrally. Thus, the CDM folder and manifest may be in different ADLS Gen2 or GitHub accounts referenced by different Linked Services.

Note: if the CDM definition file that contains the entity being referenced imports other CDM definition files, whether directly or indirectly, all the referenced files must also be available for reference by ADF at runtime. If imported files are referenced in separate folders, it is important to ensure the entire corpus, or body, of referenced CDM definitions is accessible in the linked service and that the folder structures are retained. When configuring a source or sink, you reference the root of the corpus.

Referencing the entity in the CDM folder

When writing a CDM entity to a CDM folder, if needed, ADF will create the required folders and update the manifest with the entity reference and include a resolved entity definition in the CDM folder. You can choose where the entity information is located within the folder. A common pattern is to place entities in a sub-folder. Doing this, isolates entities and makes the CDM folder easier to understand. ADF uses a sub-manifest for each entity and will place this in an entity folder if created and will write data files into separate timestamped data subfolders.

When configuring a data source to read from an entity, the location of the entity definition, the CDM folder, and the entity location are also needed (this requirement may be removed in future).

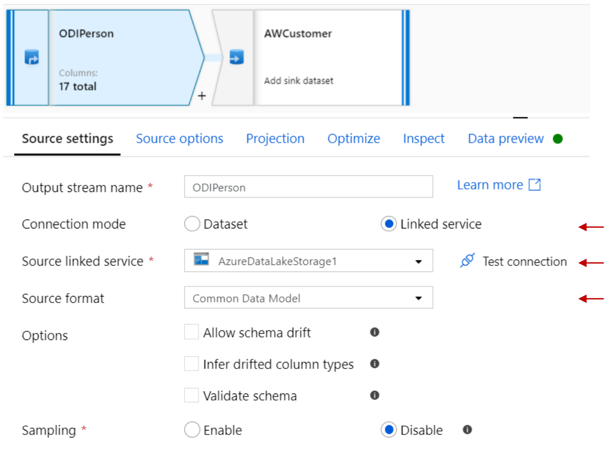
Working with model.json

Mapping data flows support reading from entities in a CDM folder described using the older model.json format. To use a model.json, reference it in place of the manifest file in the source options tab when configuring the data source options (see screenshot below). Note that a model.json file must be named exactly ‘model.json’, so there can only be one in a folder, unlike manifests, which can represent potentially overlapping subsets of the data in a CDM folder.

Example Screen Shots

The following annotated screen shots show how to configure a data source and data sink to use CDM entities. You will need to have previously configured an ADLS Gen2 Linked Service for the CDM Folder and if necessary, a separate ADLS Gen2 or GitHub Linked Service for the CDM entity definition.

Configuring a Data Source



Output stream name * ODIPerson

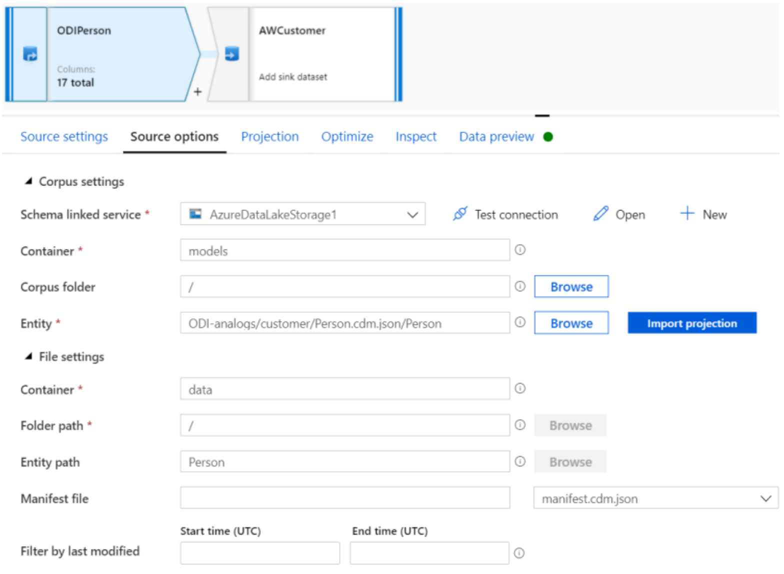
Connection mode ( ) Dataset (●) Linked service ← Select Linked Service

Source linked service * AzureDataLakeStorage1 ← Linked Service to ADLS Gen2 account containing source CDM folder

Source format Common Data Model ← Select Common Data Model

Options
- ☐ Allow schema drift
- ☐ Infer drifted column types
- ☐ Validate schema

Sampling * ( ) Enable (●) Disable



Corpus settings

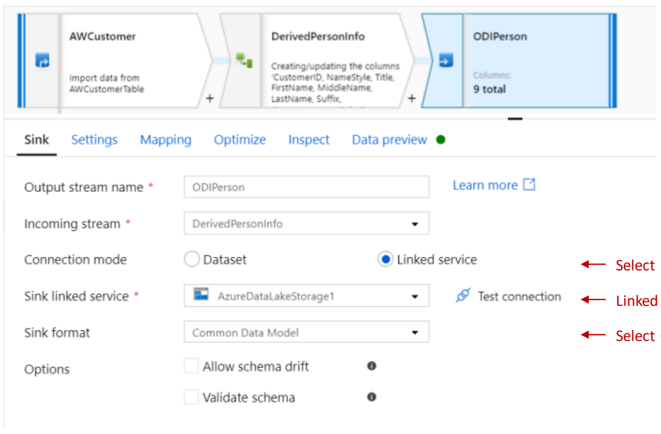
Schema linked service *	AzureDataLakeStorage1	← Linked Service to ADLS Gen2 or GitHub account with entity definition
Container *	models	← ADLS Gen2 Container containing CDM entity definition
Corpus folder	/	← Path within container to top-most folder in the CDM definition
Entity *	ODI-analogs/customer/Person.cdm.json/Person	← Full path to entity definition within corpus. Formatted as <path>/<filename.cdm.json>/<entityName>

File settings

Container *	data	← ADLS gen2 Container containing CDM folder
Folder path *	/	← Path to CDM folder within container
Entity path	Person	← Path to folder containing Entity within CDM folder
Manifest file	manifest.cdm.json	← Manifest file name (excluding manifest.cdm.json). Only needed if not 'default'. Use drop-down to select model.json

Filter by last modified — Start time (UTC) / End time (UTC)

Configuring a Data Sink


AWCustomer
Import data from AWCustomerTable
DerivedPersonInfo
Creating/updating the columns 'CustomerID, NameStyle, Title, FirstName, MiddleName, LastName, Suffix,
ODIPerson
Columns: 9 total
Sink
Settings
Mapping
Optimize
Inspect
Data preview
Learn more
Test connection

Output stream name * ODIPerson

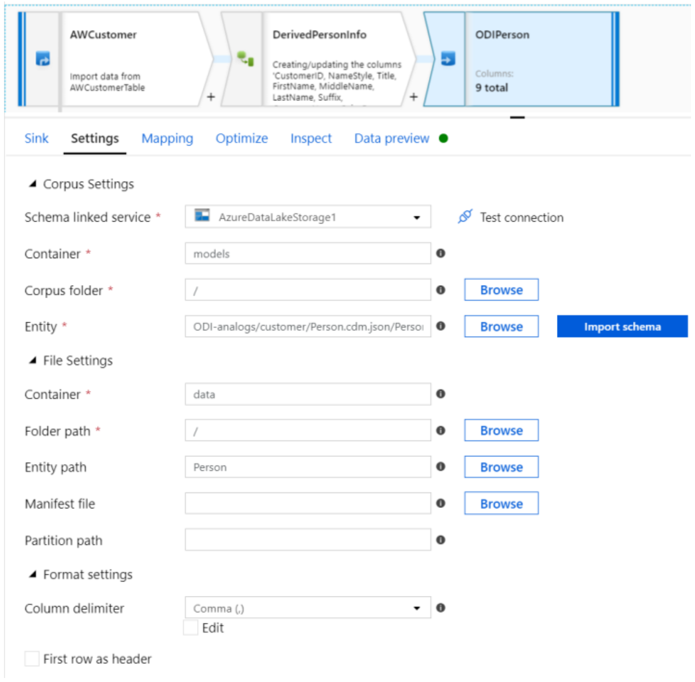
Incoming stream * DerivedPersonInfo

Connection mode ( ) Dataset [✓ Linked service] ← Select Linked Service

Sink linked service * AzureDataLakeStorage1 ← Linked Service to ADLS Gen2 account containing target CDM folder

Sink format Common Data Model ← Select Common Data Model

Options
- [ ] Allow schema drift
- [ ] Validate schema


AWCustomer
Import data from AWCustomerTable
DerivedPersonInfo
Creating/updating the columns 'CustomerID, NameStyle, Title, FirstName, MiddleName, LastName, Suffix,
ODIPerson
Columns: 9 total
Sink
Settings
Mapping
Optimize
Inspect
Data preview
Test connection
Browse
Import schema

Corpus Settings

Schema linked service * AzureDataLakeStorage1 ← Linked Service to ADLS Gen2 account containing CDM entity definition

Container * models ← ADLS Gen2 Container/File System containing CDM entity definition

Corpus folder * / ← Path within container to top-level folder in the CDM definition

Entity * ODI-analogs/customer/Person.cdm.json/Perso ← Full path to entity definition within corpus. Formatted as <path>/<filename.cdm.json>/<entity>

File Settings

Container * data ← ADLS gen2 Container/File System containing CDM folder

Folder path * / ← Path to CDM folder within container

Entity path Person ← Path to folder containing Entity within CDM folder

Manifest file ← Manifest name (excluding manifest.cdm.json). Only needed if not 'default'

Partition path

Format settings

Column delimiter Comma (,) ← Required format settings for output files. CSV only initially

- [ ] Edit
- [ ] First row as header