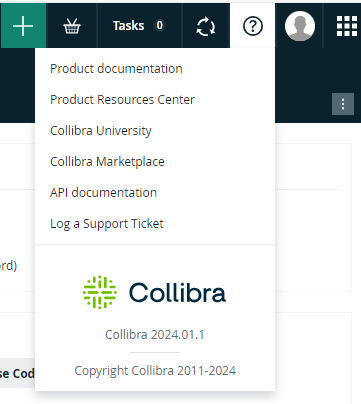
****

<https://intapi-vlab.collibra.com/docs/index.html>

<instanceURL>/docs/index.html

**REST Import API**

**Overview**

* Create or update data in bulk
  + Communities/Domains/Assets
  + Characteristics (attributes, relations, or complex relations)
* Import Methods
  + /import/json-job
  + /import/csv-job
  + /import/excel-ob
* Each method requires a file or a field
* The import will convert the file (if necessary) to a JSON format

**Import APIs**

**Comparison**

* Import API vs Core API
  + Better suited for bulk actions
  + Faster since APIs are executed in bulk
* Import APOI vs Catalog
  + Granularity of what gets imported
  + Import data Catalog doesn’t support
  + Does not allow for all Catalog Functionality
    - Profiling
    - Sampling
    - Data Classification
    - Etc.
* Note: You should only import data that you want to be managed in Collibra. Bringing too much data could transform your Collibra Platform into a Data Swamp!

**JSON File Format**

**Array of objects**

[

{command for Community 1},

{command for Community 2},

…

{command for Community N},

{command for Domain 1},

…

{command for Domain M},

{command for Asset 1},

…

{command for Asset P},

…

]

**Tips**

* When importing simultaneously import communities before domains
* If mapping between resources, import the resources first, then map them
* You can import
  + Communities
  + Domains
  + Assets
  + Relations
  + Attributes
  + Tags
  + Mappings

**Import API Resources**

**Documentation**

* <https://university.collibra.com/developer/rest/import-api-documentation>
* <instance\_URL>/docs/rest-importer/index.html

**Technical Lineage**

**Example**

* Use Collibra Data Lineage to ingest and extract technical lineage from many different sources
* What if the source is not currently supported?
* Use the import API:
  + Construct the JSON payload in accordance with the schema
  + Make a Colliibra REST Post call to /rest/2.0/import/json-job
  + Wait for the import job to finish with the Collibra REST API   
    /rest/2.0/jobs/{jobId}
  + Review imported relationships/lineage in Collibra Platform

**Scenario**

**Details**

* We are going to bring in domains, assets, and a complex relation between those assets
* In order to do this efficiently, we will use the Import API to bring in these resources simultaneously

**Task 1**

Encode our Collibra password in base64

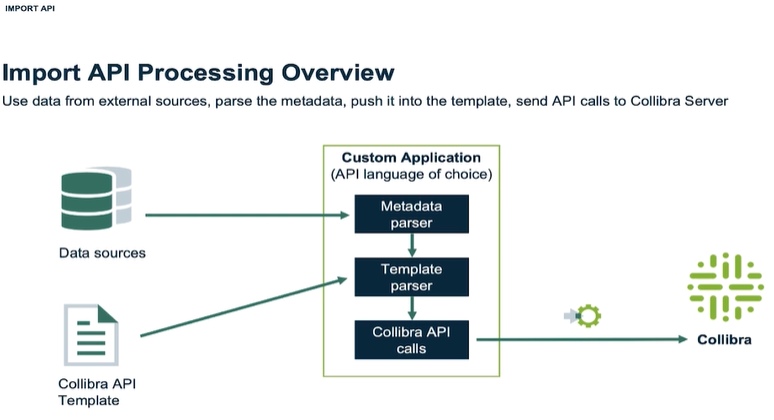
**Task 2**

Create JSON Templates

**Task 3**

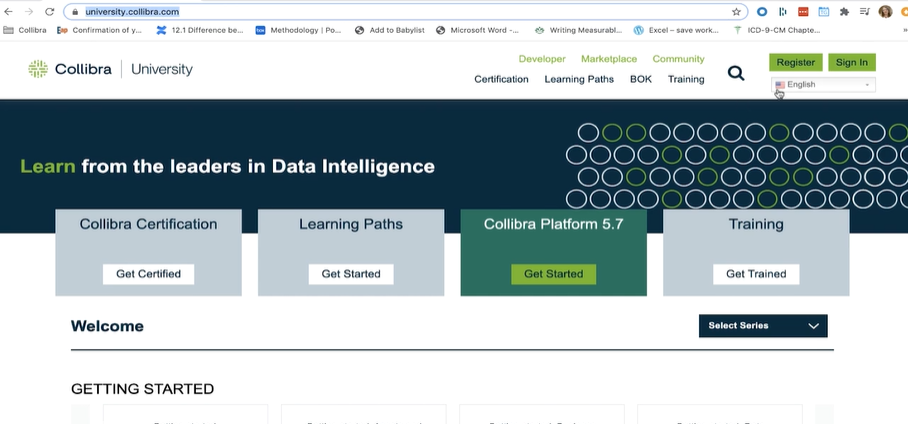
Bring in the following using Import API

* Data Asset Domain
* Mapping Domain
* 2 Data Element Assets
* 1 Mapping Specification Asset
* 1 Complex Relation between them with transformation logic

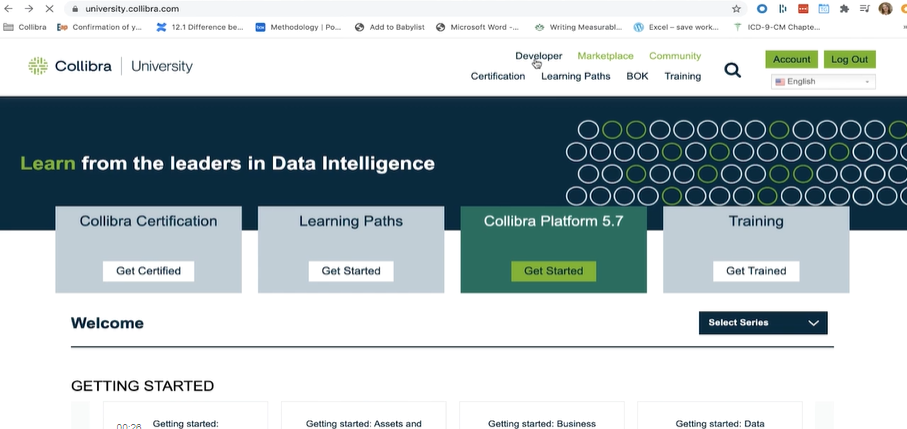


How to access – Ignore below – use [developer.collibra.com/api/rest/import-api](https://developer.collibra.com/api/rest/import-api)

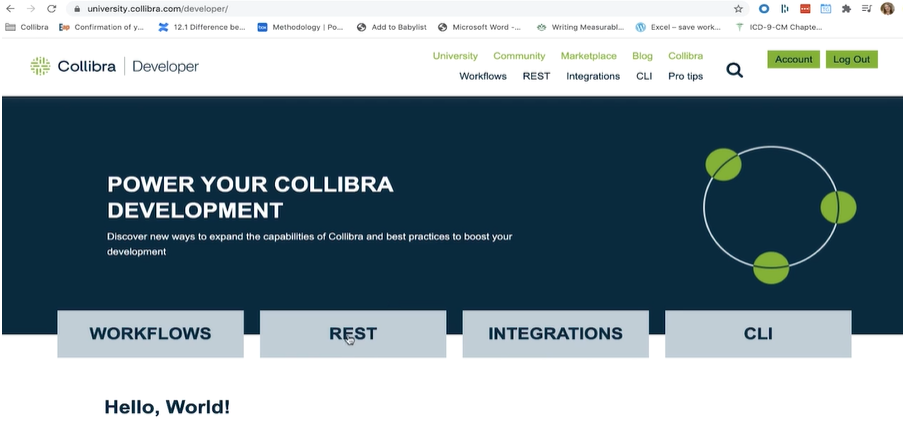
Sign in



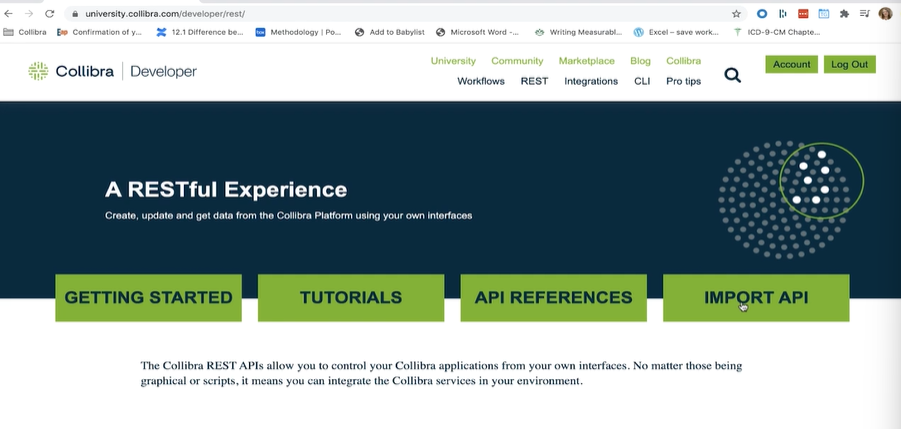
Click Developer

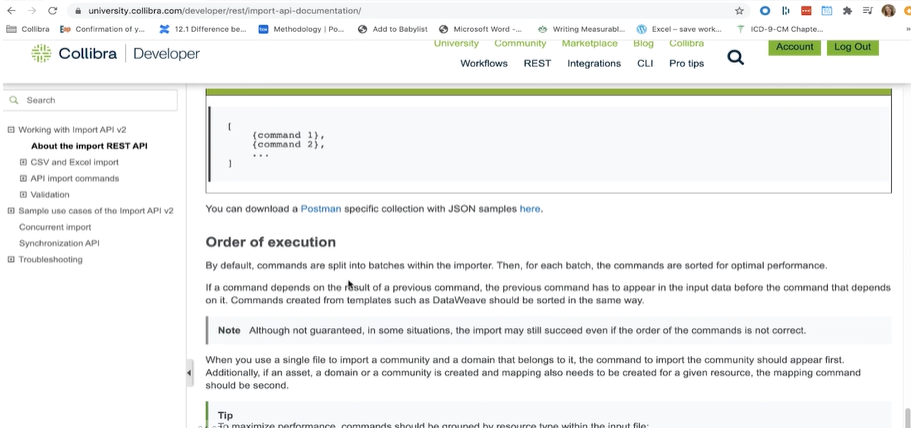


Click REST



Click Import API





**REST Call URL**

**Consists of 3 parts**

* Instance base URL (https://<your\_collibra\_url>)
* REST application path (/rest/2.0)
* End point path (/application/info)
* Example
  + https://<your\_collibra\_url>/rest/2.0/application/info

**Authentication**

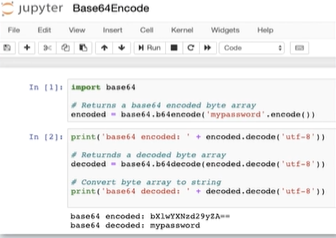
**Details**

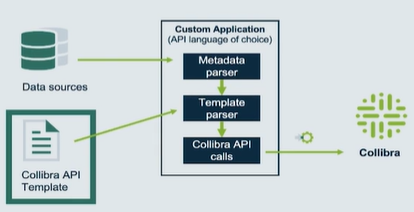
* OOTB session idle timeout is 30 minutes between API calls
* Minimally requires URL, username, and password
* We will encode our passwords using Base64

**Jupyter Notebook**

**Overview**

* Open-source web application
* Create/share live code in your language of choice
* In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection
* Execute code from the browser
* Start running a notebook server from the command line “jupyter notebook”





**JSON Templates**

We want to create

* Domains
* Assets
* Complex Relations between our assets

We will need

* Domain Template
* Asset Template
* Complex Relation Template

**Complex Relation**

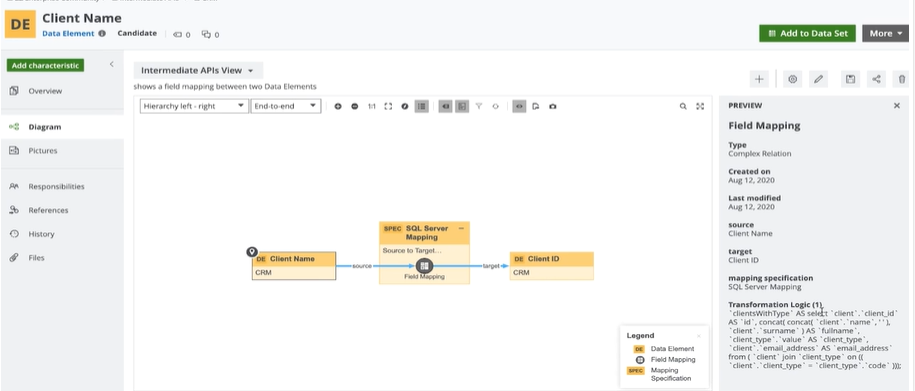
**Details**

Assets

* Client Name (Data Element)
* Client ID (Data Element)
* SQL Server Mapping (Mapping)
  + Has Transformation Logic, which is logic used to transform Client Name, Client ID

Field Mapping (Complex Relation)

* Source: Client Name
* Target: Client ID



**Asset Properties**

|  |  |
| --- | --- |
| **Asset Property** | **Description** |
| id | Uniquely identifies an existing asset in the Collibra Platform. Created automatically when creating an asset in Collibra |
| name | The full name of the new asset. Should be unique within the domain |
| displayName | (optional) The display name of an asset. If not specified, the display name will be the same as the name |

**Collibra API – POST Asset**

**response**

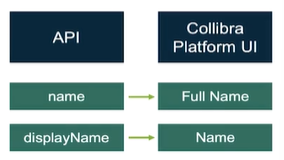


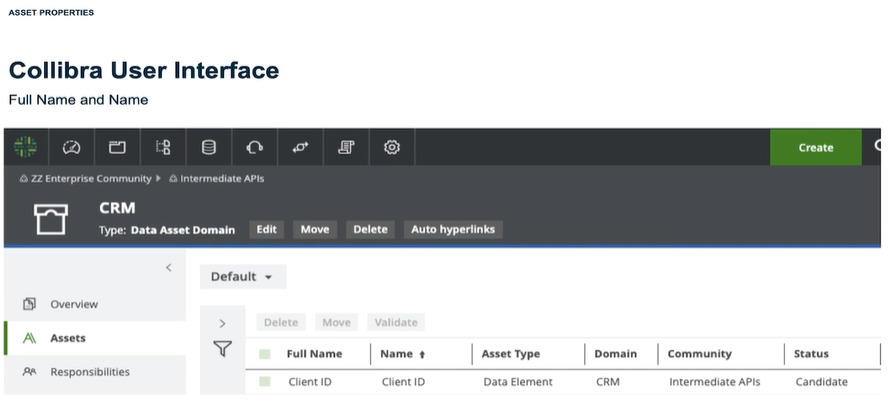
**Asset Name in the API vs. the Collibra Platform User Interface (UI)**

**API enhancement**

**(Collibra Platform version 5.4+)**

* name: This is the unique full name of the asset and is represented by **Full Name** in the UI
* displayName: This value of this argument is used as **Name** in the UI
* By doing this, application built prior to 5.4 were not affected by this change



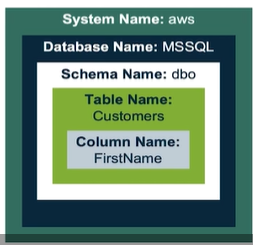


Fully Qualified Name

system + database + schema + table + column name

* Usually creates a unique value
* Use a period to separate individual components
  + Example: aws.database.schema.table.column
  + In this case . is used as a separator (any separator can be used, but should be standardized across all the imports)

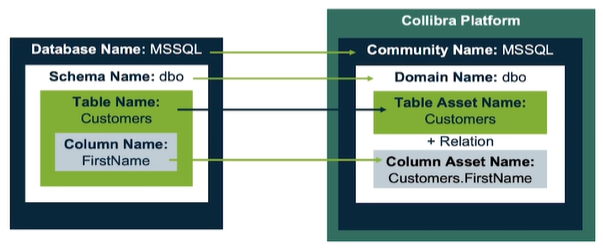
|  |  |
| --- | --- |
| Collibra Platform | Column Asset |
| name | aws.MSSQL.dbo.Customers.FirstName |
| displayName | FirstName |



**Multiple domains/communities option**

**Compound Names can lead to very long column names**

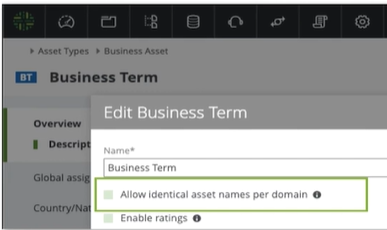
Reduce name length by using separate communities and domains based on database and/or schema

****

**Asset display name**

**Adding assets in the User Interface**

* If enabled for the asset type, you can enter a name with a different displayName
* If you don’t use displayName, the displayName will automatically get the value of the name argument
* Therse may be set differently for new installs vs. upgrades
  + New installs have this turned on for some data asset types by default
  + Upgrades will not have this turned on since it previously did not exist
* Note: the displayName is what users typically see while navigating/searching in the Collibra user interface



**Upsert Operation**

**Upsert Assets**

**Overview**

* The Import API uses the Upsert operation
* Upsert -> **Up**date or In**sert**
  + Creates new assets and characteristics simultaneously in the Collibra Platform
  + Updates existing assets in the Collibra Platform
* Requires using asset identifiers to determine which asset to create/update

Asset Name

“resourceType”: “Asset”,

“identifier”: {

“name”: “TAB\_1\_COLUMN\_1”,

“domain”: {

“name”: “Physical Domain”,

“community”: {

“name”: “DBs Community”

}

}

}

**Overview**

* Enables you to either create or update assets by using a name and location
* The name field is human readable
* It’s easy to understand what’s happening during the update process
* Useful if your assets do not have an external id

**Upsert asset by name**

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| The name field is human readable | If the colun name changes, Collibra will insert a duplicate asset, instead of updating the existing asset with the new name |
| It’s easy to understand what’s happening during the update process | The same column name can be used in multiple tables in a database |

**Asset Id**

“resourceType”: “Asset”,

“identifier”: {

“id”: “6fba073c-3e3c-4b64-a164-c74ca99e1ca8”

}

**Overview**

* Enables you to update assets by using the Collibra UUID of the asset
* Not as commonly used to identify an asset
* Id’s are more commonly used to identify asset types, domains, domain types, or communities

**Asset External Id**

“resourceType”: “Asset”,

“identifier”: {

“externalSystemId”: “EXT\_SYSTEM”,

“externalEntityId”: “EXT\_SCHEMA\_1\_ID”

},

“name”: “SCHEMA\_1”,

“domain”: {

“name”: “Physical Domain”,

“community”: {

“name”: “DBs Community”

}

}

**Overview**

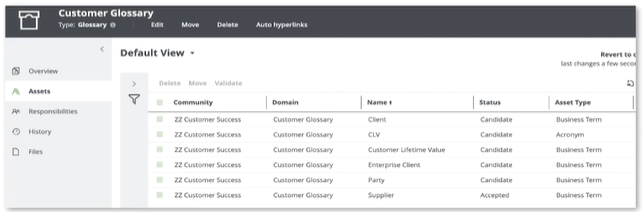
* Uniquely identifies an entity in an external system
* Either create or update assets by using ids from an external system
* Specify the External System Id and External Entity Id
* Used to find the related asset in the Collibra Platform and synchronize it with an entity from an external system, if the entity has been synchronized before
* Not visible from the Collibra User interface



**Asset location**

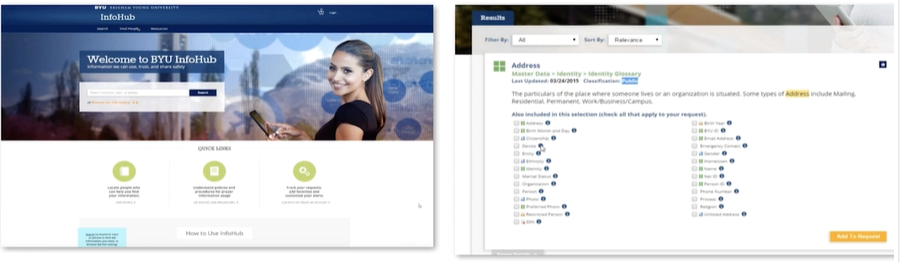
**Upsert assets to a new or existing domain/community**

* Specify the name of the domain and the name of the community
* Specify the id of the domain to move the asset to a new domain



**Search Resources**

**Web Search Portal powered by Collibra REST API**



Example Use Case – Adobe

Adobe uses Collibra to strengthen their data culture and deliver business value through deeper data understanding



Search API

Overview

* Used to create integrations with the Collibra Search Engine
* Methods
  + POST/search
  + GET/search/views
  + GET/search/views/{id}
* Search Views refer to filters, you can:
  + Get your search view
  + Copy the resulting search request
  + Use it for the Post search method
  + Modify it to suit your needs

**Post Search Query**

**Search Request Schema Examples**

**Keywords**

* (required) The search term, which can include wildcards and quotes

**Search in Fields (searchInFields)**

* A filter to refine the search results based on the occurrence of the search term in specific fields of resource type

**Search Filter (filters)**

* Refine the search results based on specific types, statuses, dates and tags of the returned resources

**Search Aggregation (aggregations)**

* (Optional) Counts the results that match the search criteria (including all filters) by specified categories

**Search Highlight (highlights)**

* Highlight options for the content that matches the search criteria
* Result matching the criteria are enclosed with tags

**Offset and Limits**

**Details**

Having too many results can hinder performance, and create a large list to sort through

**Limit**

* The number of search results to present in the response

**Offset**

* The number of first search results to skip in the response. The field provides a method to paginate the results

Together, limit and offset allow you to implement paging and to navigate through those results

**Sort fields and sort order**

**Details**

**Relevance (default)**

* Based on how close the result is to the keywords

**Name**

* The name of the returned result

**Last Modified**

* The date and time the returned result was last modified

**Ascending (asc)**

* Alphabetically ascending (default when sorting by name)

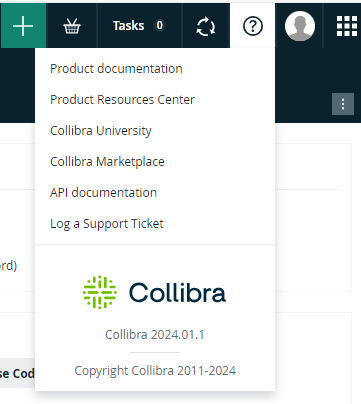
**Descending (desc)**

* Alphabetically descending (default when sorting by Relevance or Last Modified)

**Searching**

* Find a literal piece of text with “quotation marks”
  + In JSON, you will need to escape a quote (\”)
* Unsure of the name? Use wildcards
* Note: when you search in the user interface, the wild card \* is automatically added, this is different behavior than when searching using the REST Search API

|  |  |  |  |
| --- | --- | --- | --- |
| **Wildcard** | **Description** | **Example** | **Results** |
| ? | Replaces any single character | ?ar | Car, tar, bar, etc. |
| \* | Replaces any string of characters | C\*r | Car, crossbar, chair, etc. |
| - | Represents a fuzzy search, words with spelling like the search query | -Owi | Ozi, Zowie, Bowie, etc. |
| ! | Excludes words | !new car | old car, car, car travel |
| space | Space between keywords | new car | New job, car travel, new car |

****

<https://intapi-vlab.collibra.com/docs/index.html>

<instanceURL>/docs/index.html

**Update Behavior**

|  |  |
| --- | --- |
| **Resource** | **Behavior** |
| Communities | Merge |
| Domains | Merge |
| Assets | Merge |
| Asset Attributes | Set/replace |
| Asset Relations | Set/replace |
| Asset Tags | Set/replace |
| Complex Relations | Merge |
| Mappings | Merge |

Import API operations

Performs one of two operations based on the resource

Set/replace

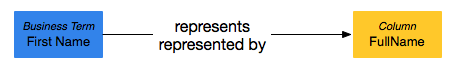
* If the resource exists with the same properties as defined in the input, no action is performed
* If the resource exists with properties other than the ones defined in the input, the resource is replaced with the one provided in the input
* If the resource does not exist, it is created

Merge

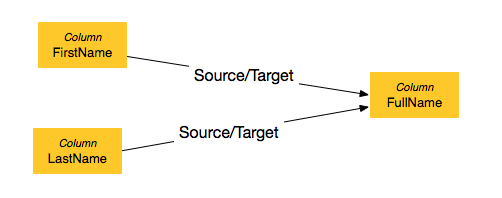
* If the resource does not exist, it is created with all the properties provided in the input
* If the resource exists, it is updated so that its properties match the ones provided in the input

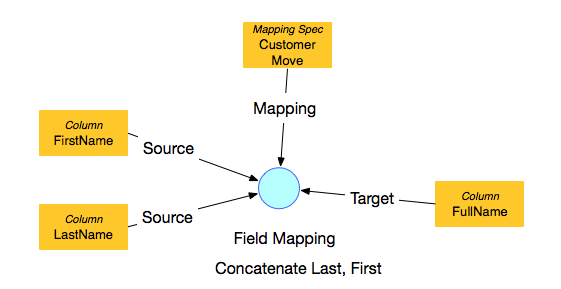
Simple vs. Complex Relationships

Relationships are a key component in Data Governance Center. One question that comes up frequently is when should you use a simple relationship and when should you use a complex relationship. The real question is what is the relationship trying to show. To answer this question you need to understand both relationship types.

**Simple Relationships**  
These are the most basic types of relationships and the most common relationship created in DGC. Take using a business term to represent a data asset. The business term can represent multiple columns but there is no confusion about what the relationship is doing since it’s just defining the data asset.  


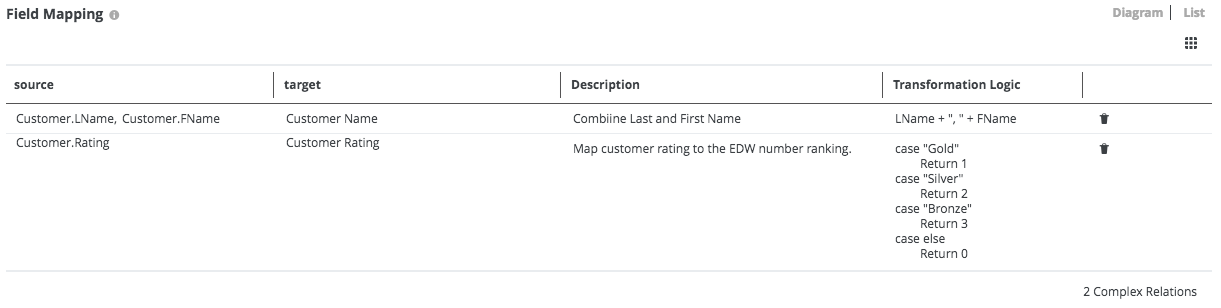
If you tried to use the simple relationship for something more complex like data movement it wouldn’t work as well. In the example below there is an issue with understanding what’s going on with the source/target relationship.

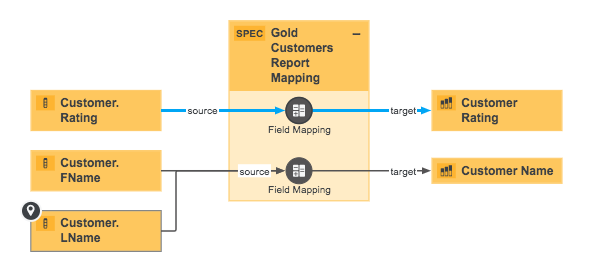
* How are FirstName and LastName combined into FullName?
* Is the data combined or is it overwritten?
* Who owns the data conversion process?
* Are other columns mapped at the same time?  
  

**Complex Relationships**  
To resolve this issue when mapping columns we can create a complex relationship. The complex relationship allows you to select multiple sources and multiple targets but one mapping. For each source to target, you can define logic used to combine the two columns into one.  


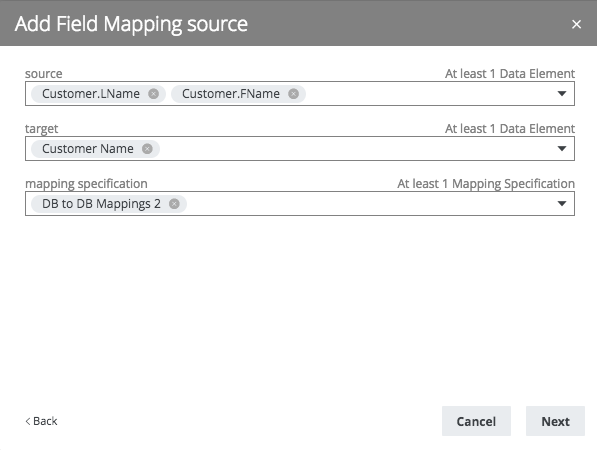
The complex relationship allows for only 1 Mapping Specification with multiple sources to target mappings. You can now assign an owner for the mapping specification and other critical information. The following is an example of what might be documented in the mapping specification.

**Description**: ETL task that moves data from Sales database to the Reporting warehouse.  
**Scope**: SSIS job that runs on SQL Server.  
**IT Requirements**: Run Daily to pull updated or new records from sales within the last 24 hours.  
**Owner**: Jane Doe  
**Technical Steward**: John Doe  
**Stakeholders**: Tom Sales, Bill Boss

The screenshot below from a mapping specification in DGC shows how multiple columns are mapped from the source to the target in a single mapping. Notice the Description of how the transformation is done along with a transformation logic in the form of code snippets.  


When you trace the column you can now visualize that the mapping has multiple field level conversions that occur at the same time.  
**NOTE**: This is a custom visualization where the mapping boxes the field mappings to better illustrate that multiple fields flow through the mapping.  


**Creating Complex Relationships**:  
Another key difference between simple and complex relationships is how they are created.

* Through the UI you can add the relationship from a column (source or target) or at the mapping specification. You select Multiple Source data assets, multiple target data assets, and one mapping specification.  
  
* Simple relationships can be created when you import the assets. Complex relationships have a separate import process that requires you to specify the complex relationship you want to create prior to mapping the data.
* Complex Relationships cannot be shown on domain views (i.e. it is not possible to bulk edit complex relationships).
* You can also use MuleSoft and Collibra Connect to load complex relationships but you need to setup file import process.
* You can use MuleSoft and rest API calls to create complex relationships.
* The complex event listener, available on our marketplace, can be set up in MuleSoft to act as a complex relationship loader.

REST Import API Documentation

[developer.collibra.com/api/rest/import-api](https://developer.collibra.com/api/rest/import-api)

JSON Validation

[JSON Online Validator and Formatter - JSON Lint](https://jsonlint.com/)