**Application Program Interface (API)**

Definition

* A set of routines, protocols, and tools used to build software applications by specifying how software components should interact
* A messenger that takes requests, tells the system to do, and then returns a response

**Why use APIs**

Benefits

* Allows you to perform tasks in a much more automated fashion
  + E.g. script tasks in workflows
* Getting information into and out of the Collibra Platform
  + E.g. REST calls to POST/GET information
* Ability for external systems to communicate for easier integrations
  + E.g. Integration with a Data Quality Tool via Connect

**Collibra API options**

* Java
  + Used when authoring Collibra workflows
  + Changes are initiated from the Collibra Platform
* REST
  + Used for 3rd party integrations
  + Outside applications call to the Collibra Platform

**API documentation**

<Instance URL>/docs.index.html

Sys admins can navigate from Collibra Platform

**REST – Representational State Transfer**

Details

* Resource-based
* Representations transfer JSON or XML to represent data objects and attributes
* Messages use HTTP methods (GET, POST, PUT, and DELETE)
* Stateless server does not store any state about the client session on the server
* Each request must contain all necessary information

**Client Requests**

Include

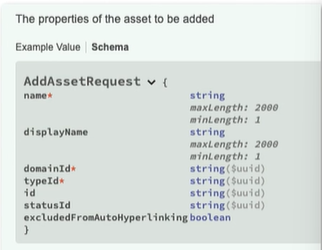
* Endpoint (URL, specifies the resource to act upon)
* Method (action to perform)
* Headers (request/respond with meta-data)
* Body (parameters to send) for applicable methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| Get | Retrieve resource information |
| Post | Create new resource |
| Put | Update existing resource or create new resource |
| Patch | Update specific fields of an entity |
| Delete | Delete resource |

**Schema**

Contains request parameters

* Required parameters indicated with a red asterisk
* Example Value can be viewed for reference
* Schema is different per API call



**JSON (JavaScript Object Notation)**

KEY-value pairs

{

“name”: “Paid in Full”,

“domainId”: “a0003ec8-30dd-4ca4-9e73-4f777a0cd7de”,

“typeId”: “0000000000-0000-0000-0000-00000000011001”

}

**HTTP server response**

After the request is made, the server responds with

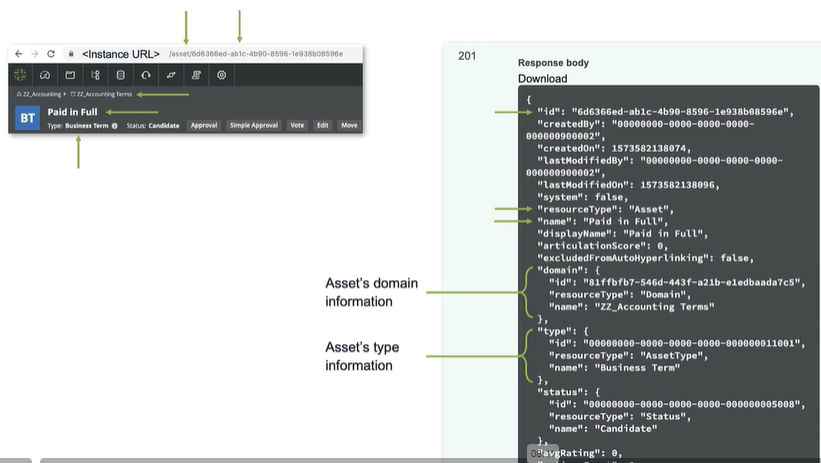
* A three-digit status code
* A list of headers
* A body

|  |  |  |
| --- | --- | --- |
| **Status Code** | **Description** | **Example** |
| 1xx | Informational | First part of a request has been received |
| 2xx | Success | Asset was created successfully |
| 3xx | Redirection | Asset has moved |
| 4xx | Client error | User is not logged in, resource not found |
| 5xx | Server error | Server could not complete the request due to an unexpected error |

HTTP server response

Once created, capture each of the following per resource

* Resource Name
* Resource Type
* Resource ID



**Scenario**

Via REST, each of us will POST:

A community named

* <initials>\_Accounting

A Domain

* <initials>\_Accounting Terms

Assets

* Paid in Full
* Partial Payment
* Unpaid
* PIF
* PP
* UP

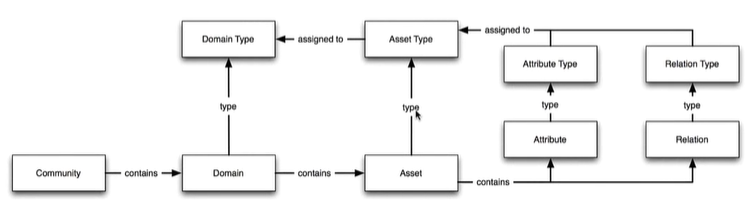
Attributes

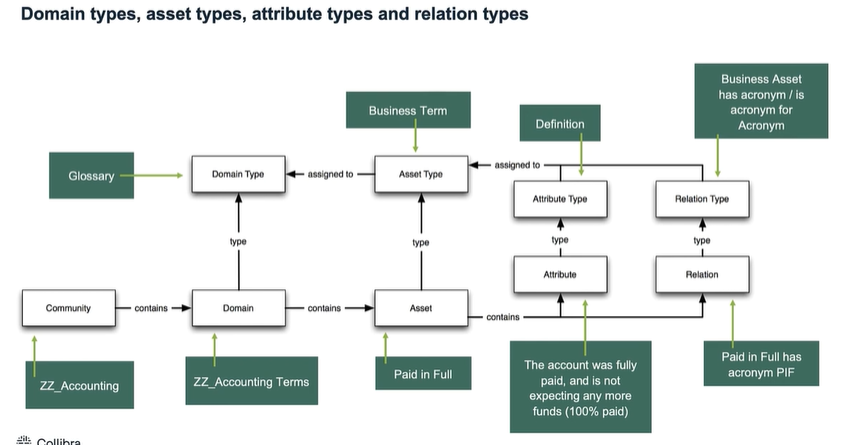
* Add definitions for the terms

Relations

* Relate terms with their acronyms

**Domain types, asset types, attribute types and relation types**

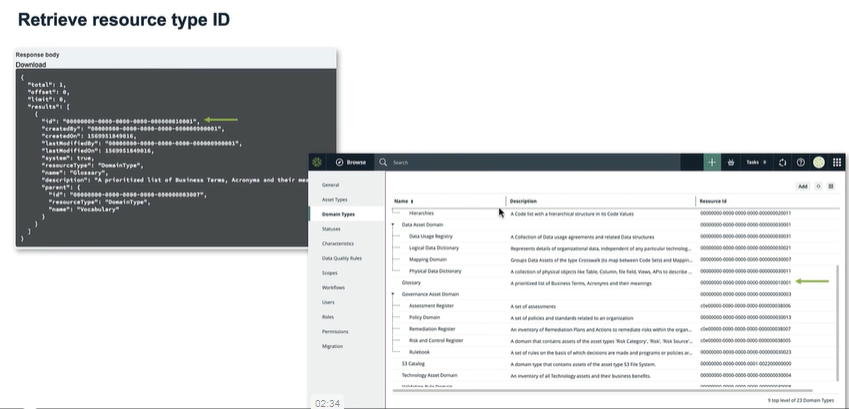




**Domain Type ID**

**Search Options**

* Collibra Platform
  + Search in Settings > Domain Types
  + If necessary, add the Resource ID column
* API documentation
  + REST Core API Version 2 > GET /domainTypes
  + Filter API response by name



**Asset Type ID**

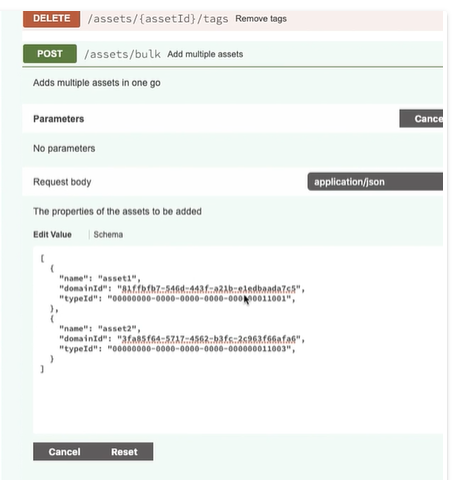
**Search Options**

* Collibra Platform
  + Search in Settings > Asset Types
  + If necessary, add the Resource ID column
* API documentation
  + REST Core API Version 2 > GET /assetTypes
  + Filter API response by name

**POST in bulk**

**Request Details**

* Add multiple resources simultaneously
* The bulk request is contained in an array [ ]
* Each resource is in brackets { }, separated by commas
* The same resource of varying types can be added by specifying the typeId
  + Example, when posting Assets in bulk you can post Business Terms and Acronyms in the same API call



**Attribute Type ID**

**Search Options**

* Collibra Platform
  + Search in Settings > Attributes
  + If necessary, add the Resource ID column
* API documentation
  + REST Core API Version 2 > GET /attributeTypes
  + Filter API response by name

**Attributes and Attribute Kinds**

**“Value”: { }**

* When adding attributes, you give a value for the attribute
* A placeholder is used since the expected Data Type depends on the Attribute Type
* Possible Data Types with examples
  + String – “some text in quotes”
  + Integer – 7
  + Boolean – true
* The Attribute Type’s Attribute Kind determines which Data Type is needed



**Relation Type ID**

**Options to search for the role/co-role**

* Collibra Platform
  + Search in Settings > Characteristics > Relations
* API documentation
  + REST Core API Version 2 > GET /relationTypes
  + Filter API response by role or coRole

**Comments**

* Comment on a Resource
  + Specify the Resource Id
  + Specify the Resource Type
* Reply to an existing comment
  + Specify the parentId

**Java API**

**Used in Workflow script tasks**

* The script task executes the provided script when the workflow arrives at the task
* A script language must be specified, as well as a script, otherwise the workflow will not deploy
* We recommend the groovy script-language

Java tips and tricks

|  |  |
| --- | --- |
| **Code** | **Description** |
| // | Comments out a single line of code |
| /\* followed by \*/ | Comments out multiple lines of code |
| ${ } | Evaluate an expression |
| [ ] | Indicates a list/array |
| [:] | Indicates a map |
| def = variable | Creates a referenceable variable |

**Request objects**

**Manipulating objects**

There are request objects to

* Create asset/attribute/etc.
* Set attribute/relation/tags
* Get asset/attribute/etc.
* Remove tags/users
* Export files
* The request objects are required in order to perform the associated method

**Builders**

**Details**

* Used to create and configure objects via the build() method
* Returns the fully constructed object
* Example: AddAssetRequest.Builder

**Add asset request builder parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Java Object Type | Required? | Description |
| displayName | string | N | Sets the display name of the new asset |
| domainId | UUID | Y | Sets the id of the domain that the new asset should be added to |
| name | string | Y | Sets the full name of the new asset |
| typeId | UUID | Y | Sets the id of the asset type of the new asset |

UUID -> we will have to convert from a string to a UUID

**Example**

.name(“My asset name”)

.domainId(string2Uuid(“Domain id”)

.typeId(string2Uuid(“asset type id”)

.build()

Formatting a script task

|  |  |
| --- | --- |
| **Steps** | **Example** |
| Use corresponding API | AssetAPI |
| Convert to camelCase | assetApi |
| Copy the method and its parameters to understand the format | assetApi.addAsset(AddAssetRequest addAssetRequest) |
| Import the Request Object | Import com.collibra.dgc.core.api.dto.instance.asset.AddAssetRequest |
| Build the request with the name of the builder | assetApi.addAsset(AddAssetRequest.builder() |
| Add parameter from the builder | .name(String name).domainId(UUID domainId).typeId(UUID typeId) |
| Finish building | .build() |

**Script Example**

Import com.collibra.gc.core.api.dto.instance.asset.AddAssetRequest

//def asset => Define a variable to reference the object once it is created

def asset =assetApi.addAsset(AddAssetRequest.builder()

.name(“Paid in full”

.domainId(string2Uuid(“301783ef-dd1d-4be8-a5a6-b7a140dfab15”)

.typeId(string2Uuid(“00000000-0000-0000-0000-000000011001”)

.build()

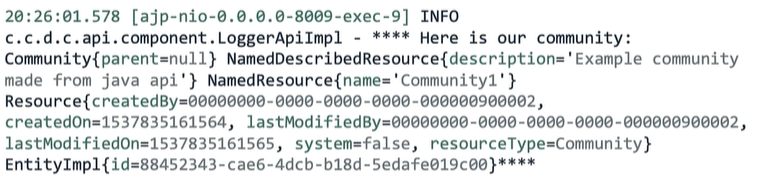
)

**LoggerApi**

Adds information to the log file

* Use the logger API to see information about the created objects
* Use easily searchable phrases such as
  + \*\*\*\*This is my new asset\*\*\*\*
* Leverage variables to reference the objects in the log file
* Ass the parameters will display in the log file (like the REST responses)

**Log file entry example**



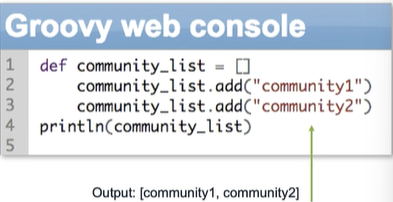
**Build request objects from an array**

**Create multiple objects simultaneously**

Strategy

* Define a variable to represent the array
* Add a list of objects, with the required builder methods/parameters
* Use the builder against the array

Java Array



Java Class ArrayList <E>

Java Platform Standard Edition 8 documentation

<https://docs.oracle.com/javase/8/docs/api/index.html?java/util/List.html>

|  |  |
| --- | --- |
| **Modifier and type** | **Method and description** |
| Boolean | add(E e) Appends the specified element to the end of this list |
| E | get(int index) Returns the element at the specified position in this list |