The Datum360 CLS360 Snapshot Json structure.

This end point returns a large Json structure.

var url = $"{cls360url}api/domains/{DomainHandle}/snapshot/?version={Version}";

The top level objects are

A group of text on a white background

Description automatically generated

Domain

A computer code with many letters and numbers

Description automatically generated

Schema

A close up of numbers

Description automatically generated

CL4CL is an Array of Objects

This is an example of one of the KEY items of information the Template Functional

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

Classes are structurally very similar to CL4CL, they are in fact an Instance of a given Template object.

This is an example of a Instance of a functional template

A screenshot of a computer

Description automatically generated

A close up of a document

Description automatically generated

A screenshot of a computer code

Description automatically generated

Functional Class Pipe Mapping.Attributes ..

The GUIDs or Hdls in this array map to other objects in the classes array of objects

A close up of text

Description automatically generated

A white sheet with text on it

Description automatically generated

Classes grouped by the Object Type would typically look like this, but other Object Type can be created, in CL4CL

A screenshot of a computer code

Description automatically generated

A Functional ObjectType represents a Class/Or Asset Type like a PIPE or PUMP.

A Information Attribute represents an Attribute that could be used by a Functional ObjectType for example.

"chemical cleaning required"

A Measure Attribute has further Associations/Relationships to UoM/UoM Group, as well as there is a Unit concept to the Value, like Length in Meters or Kilometres etc..

**000 - Data Standards Manager Version 69**

ObjectType is from

A close-up of text

Description automatically generated

CL4CL By ObjectType

Key={"ObjectType":"Alias"} Count=34

Key={"ObjectType":"Associations"} Count=6

Key={"ObjectType":"Data"} Count=6

Key={"ObjectType":"Mapping"} Count=23

Key={"ObjectType":"Metadata"} Count=18

Key={"ObjectType":"Template"} Count=19

Classes By ObjectType

Key={"ObjectType":"Attribute Group"} Count=4

Key={"ObjectType":"Discipline"} Count=4

Key={"ObjectType":"Equipment Item"} Count=2

Key={"ObjectType":"Functional"} Count=20

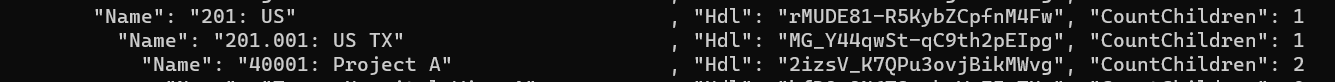
Key={"ObjectType":"Information Attribute"} Count=10

**Object Mapping ..**

An object can be linked to other objects in several ways, in the Mapping object.

Mapping is logically a dictionary where the Key is the type of Map “Superclass” and the value is an Array of Hdls.

Superclass is an array of Parent object Handles; an object CAN have multiple parents. Example (3)(2)(1)



(1)

A screenshot of a computer

Description automatically generated

(2)

A screenshot of a computer

Description automatically generated

(3)

A screenshot of a computer

Description automatically generated

To walk the Superclass hierarchy from a top-level object down you can use the following pattern.

WalkChildren is a simple recursive walk of the Superclass array of objects.

A computer screen shot of a code

Description automatically generated

The Mappings do not necessarily create a hierarchy, but just imply a relationship to.

A close up of a code

Description automatically generated

A screenshot of a computer code

Description automatically generated

"40001: Project A" is a Functional object, the Superclass links the hierarchy, Mapped To Physical a link to object of “Equipment Item”

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

This is the output from DumpObject for the above object.

A computer screen with white text

Description automatically generated

function DumpObject (item, indent) {

let pad = ''

pad = pad.padStart(indent \* 2, ' ')

console.log(`${pad}Name:[${GetObjectName(item)}] ObjectType: [${GetObjectType(item)}]`)

var keys = Object.keys(item.Mapping)

keys.forEach((key) => {

const m = item.Mapping[key]

if (m !== undefined && m.length > 0) {

console.log(`${pad} Map:[${key}] HasCount: [${m.length}]`)

m.forEach((hdl) => {

const o = MapClasses.get(hdl)

console.log(`${pad} MappedName:[${GetObjectName(o)}] MappedObjectType: [${GetObjectType(o)}]`)

})

}

})

}

The Mappings are configured in CL4CL Template Functional and the Object Type it maps is defined in the CL4CL Mapping, here the Mapping “Mapped To Physical” is mapping object from the Template Type “Equipment Item”

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The CL4CL Object for Mapped To Physical below ..

A screenshot of a computer code

Description automatically generated