Stream-X API

This document explains how works the WSG Web API to upload or download data.

The S3MX service is first intended as a prototype of streaming Reality Data starting with 3MX data. In principle it is a general file server. Since 3MX files are split into many components named and indexed relative to the root, fetching and caching individual files is sufficient to obtain a virtual streaming.

S3MX is a Bentley CONNECT service which implies information is uploaded or downloaded through a simple HTTPS based REST API. This API has a structure imposed by the underlying technology WSG that manages transfers and preparation of data.

Access to information is based on an http request.

Here is an example of such request:

<https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/S3MX/Document/Production_Graz_3MX~2FScene~2FProduction_Graz_3MX.3mx/$file>

The components of this http request can be decomposed as follows:

|  |  |
| --- | --- |
| Protocol | https: |
| Server | s3mxcloudservice.cloudapp.net |
| Port | 443 |
| WSG mandatory part | All WSG request have the WSG version number. Here it is /v2.3 |
| WSG API identification | *Repositories* indicates the main WSG API. |
| Repository name | A server may publish more than one repository. *S3MXECPlugin—Server* is the name of our repository. |
| Schema | *S3MX* is a simple schema that defines two classes which are *Folder* and *Document*. A document is a file. |
| Class name | As I said there are two classes in the schema. This request accesses a *Document* object |
| Instance Id | *Production\_Graz\_3MX~2FScene~2FProduction\_Graz\_3MX.3mx*  We will get back to the nature of this identifier |
| Parameters | There are a variety of parameters that can be added to a request that enable to filter out properties for example. The parameter here is */$file*. This parameter indicates that the content of the document is requested instead of its properties. |

## Authentication

All CONNECT services require that a valid CONNECT (also called IMS) token be provided in the header of the request. No request will be answered if a valid token is not provided. To obtain the token access to the local computer Bentley Connection Client must be done and an authorized user must be logged-in the session.

Obtaining of the Bentley Connection Client token depends on the context or coding language and will not be explained here.

## Schema

A schema defines classes. In our case we only have two which are **Document** and **Folder** and all their properties. Here is the complete S3MX schema:

**IMPORTANT NOTE: The Schema name and Schema content may change for the final version since we are trying to normalize Schema or at least class definition throughout various file publishing CONNECT Services.**

<?xml version="1.0" encoding="utf-8"?>

<ECSchema schemaName="S3MX" nameSpacePrefix="S3MX" version="1.0" xmlns="http://www.bentley.com/schemas/Bentley.ECXML.2.0">

<ECSchemaReference name="Bentley\_Standard\_CustomAttributes" version="01.00" prefix="bsca" />

<ECClass typeName="Document" isDomainClass="True">

<ECProperty propertyName="Alias" typeName="string" />

<ECProperty propertyName="ContainerName" typeName="string" />

<ECProperty propertyName="DocumentName" typeName="string" />

<ECProperty propertyName="DocumentId" typeName="string" />

<ECProperty propertyName="FolderId" typeName="string" />

<ECProperty propertyName="FileExists" typeName="string" />

<ECProperty propertyName="AccessUrl" typeName="string" />

<ECProperty propertyName="ProjectId" typeName="string" />

<ECProperty propertyName="RootDocument" typeName="string" />

<ECProperty propertyName="ContentType" typeName="string" />

<ECProperty propertyName="ModifiedTimeStamp" typeName="string" />

<ECProperty propertyName="CreatedTimeStamp" typeName="string" />

<ECProperty propertyName="ModifiedBy" typeName="string" />

<ECProperty propertyName="CreatedBy" typeName="string" />

<ECProperty propertyName="Size" typeName="string" />

<ECProperty propertyName="Path" typeName="string" />

</ECClass>

<ECClass typeName="Folder" isDomainClass="True">

<ECProperty propertyName="FolderName" typeName="string" />

<ECProperty propertyName="ProjectId" typeName="string" />

<ECProperty propertyName="ParentFolderId" typeName="string" />

<ECProperty propertyName="RootFolder" typeName="string" />

<ECProperty propertyName="ContentType" typeName="string" />

<ECProperty propertyName="ModifiedTimeStamp" typeName="string" />

<ECProperty propertyName="ModifiedBY" typeName="string" />

<ECProperty propertyName="CreatedTimeStamp" typeName="string" />

<ECProperty propertyName="CreatedBy" typeName="string" />

<ECProperty propertyName="Path" typeName="string" />

</ECClass>

</ECSchema>

All properties are strings.

## Instance Id for Documents

Since 3MX relies upon designated files named relative to a root document we needed a way to compute the Instance Id based on expected file names including path. The Instance Id here is a simple pseudo-URL encoded of the full path name of the document to be accessed.

The Instance Id ***Production\_Graz\_3MX~2FScene~2FProduction\_Graz\_3MX.3mx***is automatically decoded by WSG to **Production\_Graz\_3MX/Scene/Production\_Graz\_3MX.3mx**

Notice that WSG uses the **~** symbol instead of the usual **%.** We suppose this is to prevent accidental decoding by URL interpretation engines.

The instance Id identifies the location of the ***Production\_Graz\_3MX.3mx*** file in our virtual file structure. This file once downloaded and interpreted can now be used to display data or to find other resolutions like requesting ***Data/Production\_Graz\_3MX.3mxb***which is relative to the path of this initial 3mx file. The Instance Id of this other document can be computed by removing the name of the ***Production\_Graz\_3MX.3mx*** and appending the relative file name required yielding

***Production\_Graz\_3MX/Scene/Data/Production\_Graz\_3MX.3mxb***

Once pseudo-URL encoded we obtain:

***Production\_Graz\_3MX~2FScene~2FData~2FProduction\_Graz\_3MX.3mxb***

This is the Instance Id of the next file we will need as data root. Once the new http request is built we obtain:

[https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/S3MX/Document/Production\_Graz\_3MX~2FScene~2FData~2FProduction\_Graz\_3MX.3mxb/$file](%20https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/S3MX/Document/Production_Graz_3MX~2FScene~2FData~2FProduction_Graz_3MX.3mxb/$file)

There is a limit or 1024 characters for Instance Id which should be sufficient for most file tree structure.

**IMPORTANT NOTE: The Instance Id pattern may change for the final version since we are trying to establish ways to store data according to an enterprise or project pattern. This should however only affect the start of the Instance Id (root or first folder). The way to modify the remainder should remain as explained here.**

## Accessing Properties

When the /$file postfix is removed from the request, instead of the binary content of the file, the server sends the properties pertinent to the object referenced:

So the query  [https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/S3MX/**Document**/Production\_Graz\_3MX~2FScene~2FProduction\_Graz\_3MX.3mx](https://NAOU10087QBC:54321/ws/v2.3/Repositories/S3MXECPlugin--Server/S3MX/Document/Production_Graz_3MX~2FScene~2FProduction_Graz_3MX.3mx)

Without the terminal **/$file** returns a JSON looking like:

{

"instances": [

{

"instanceId": "Production\_Graz\_3MX/Scene/Production\_Graz\_3MX.3mx",

"schemaName": "S3MX",

"className": "Document",

"properties": {

"Alias": null,

"ContainerName": null,

"DocumentName": null,

"DocumentId": null,

"FolderId": null,

"FileExists": null,

"AccessUrl": null,

"ProjectId": null,

"RootDocument": null,

"ContentType": null,

"ModifiedTimeStamp": null,

"CreatedTimeStamp": null,

"ModifiedBy": null,

"CreatedBy": null,

"Size": null,

"Path": null

},

"eTag": "\"c//gG3kE4xgn8/5aLE0fQJJyY00=\""

}

]

}

Note that this feature is available for both classes: **Document**, **Folder**

## Traversing or Browsing Through the File Structure

Browsing through the file structure is possible through the NavNode API. There are 2 different calls to be done.

1. GET: [https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/**Navigation**/NavNode/](https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/Navigation/NavNode/ECObjects--S3MX-Folder-Production_Graz_3MX~2F/NavNode/) is for the root node
2. GET: [https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/**Navigation**/NavNode/ECObjects--S3MX-Folder-Production\_Graz\_3MX~2F/NavNode/](https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/Navigation/NavNode/ECObjects--S3MX-Folder-Production_Graz_3MX~2F/NavNode/) is for a single NavNode

Note that we are not using any Schema (S3MX) here. Instead we are using the **Navigation** module which give us the possibility to find what we are looking for. To drill down into sub-directories, you should replace the Instance Id in the URL by the Instance Id of the desired instance (url encoded and replacing % with ~).

e.g.:

1. GET **Root**:   
   [https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/Navigation/NavNode/](https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/Navigation/NavNode/ECObjects--S3MX-Folder-Production_Graz_3MX~2F/NavNode/)
2. GET **ECObjects--S3MX-Folder-Production\_Graz\_3MX~2F** NavNode: [https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/Navigation/NavNode/**ECObjects--S3MX-Folder-Production\_Graz\_3MX~2F**/NavNode/](https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/Navigation/NavNode/ECObjects--S3MX-Folder-Production_Graz_3MX~2F/NavNode/)
3. GET **ECObjects--S3MX-Folder-Production\_Graz\_3MX~2FScene~2F** NavNode: [https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/**Navigation**/NavNode/**ECObjects--S3MX-Folder-Production\_Graz\_3MX~2FScene~2F**/NavNode/](https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/Navigation/NavNode/ECObjects--S3MX-Folder-Production_Graz_3MX~2FScene~2F/NavNode/)

The request GET <https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/Navigation/NavNode/> being the request for the **root** returns a JSON looking like:

{

"instances": [

{

"instanceId": "**ECObjects--S3MX-Folder-Production\_Graz\_3MX~2F**",

"schemaName": "Navigation",

"className": "NavNode",

"properties": {

"Key\_TypeSystem": "ECObjects",

"Key\_SchemaName": "S3MX",

"Key\_ClassName": "Folder",

"Key\_InstanceId": "Production\_Graz\_3MX/",

"Label": "Folder",

"Description": "Production\_Graz\_3MX",

"CollapsedImageID": "",

"ExpandedImageID": "",

"HasChildren": false,

"Selectable": true,

"IsEditable": false,

"IsFileBacked": false,

"IsCheckBoxEnabled": true,

"IsCheckBoxVisible": false,

"IsChecked": false,

"IsDragEnabled": true

},

"eTag": "\"Cu+2RVBA9cxOtHBZUjLksReYIFs=\""

},

{

"instanceId": "**ECObjects--S3MX-Folder-Production\_Helsinki\_3MX~2F**",

"schemaName": "Navigation",

"className": "NavNode",

"properties": {

"Key\_TypeSystem": "ECObjects",

"Key\_SchemaName": "S3MX",

"Key\_ClassName": "Folder",

"Key\_InstanceId": "Production\_Helsinki\_3MX/",

"Label": "Folder",

"Description": "Production\_Helsinki\_3MX",

"CollapsedImageID": "",

"ExpandedImageID": "",

"HasChildren": false,

"Selectable": true,

"IsEditable": false,

"IsFileBacked": false,

"IsCheckBoxEnabled": true,

"IsCheckBoxVisible": false,

"IsChecked": false,

"IsDragEnabled": true

},

"eTag": "\"pk5kcPKbzDry83pvVz5vzd1KV2g=\""

}

]

}

This JSON indicates that there are two instances at the root and provides their description. In this answer you can consider that the instance is of class NavNode from the Navigation built-in schema. The NavNode object provides information about the class and instances it contains.

The instanceId of our first NavNode instance is: **ECObjects--S3MX-Folder-Production\_Graz\_3MX~2F** indicating it is an **ECObject** that the schema is **S3MX** the class name is **Folder** and **Production\_Graz\_3MX~2F** is the encoded instance ID. This is a generated id that is based on the information contained in the properties of the NavNode. We suggest that the properties be used since the InstanceId pattern may be subject to change in the future. This instanceId is nevertheless essential to navigate in the tree structure.

In order to access the referenced instance, you can use the following fields to create the normal access API

From:

"Key\_SchemaName": "S3MX",

"Key\_ClassName": "Folder",

"Key\_InstanceId": "Production\_Graz\_3MX/",

You can create

<https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/S3MX/Folder/Production_Graz_3MX~2F>

That will return the properties of the folder including contained Document or Folder.

NOTE: The Key\_InstanceId is not pseudo-URL encoded so you must do it to build this request.

Navigation into sub-folder is performed by using the instanceId of the NavNode

<https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/Navigation/NavNode/ECObjects--S3MX-Folder-Production_Graz_3MX~2F/NavNode/>

Will return

{

"instances": [

{

"instanceId": "ECObjects--S3MX-Folder-Production\_Graz\_3MX~2FScene~2F",

"schemaName": "Navigation",

"className": "NavNode",

"properties": {

"Key\_TypeSystem": "ECObjects",

"Key\_SchemaName": "S3MX",

"Key\_ClassName": "Folder",

"Key\_InstanceId": "Production\_Graz\_3MX/Scene/",

"Label": "Folder",

"Description": "Scene",

"CollapsedImageID": "",

"ExpandedImageID": "",

"HasChildren": false,

"Selectable": true,

"IsEditable": false,

"IsFileBacked": false,

"IsCheckBoxEnabled": true,

"IsCheckBoxVisible": false,

"IsChecked": false,

"IsDragEnabled": true

},

"eTag": "\"EzK3DuykwDjtJ8eSTnq5E0jEmEI=\""

}

]

}

This way it is possible to navigate in the tree-like structure to browse through the file system.

## Uploading information

In order to upload data, you must first create a new instance of an object by POSTing a JSON fragment to the server. Afterward, once the object is created the file can be transferred.

Here’s the first Http Request needed (handshake)

**PUT** <https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/S3MX/Document/Production_Graz_3MX~2FScene~2FProduction_Graz_3MX.3mx/$file>

Required headers:

1. Authorization: Token xyz [Replace xyz by a valid IMS token]
2. Content-Disposition: attachment; filename="**Production\_Graz\_3MX.3mx**" [Replace it by the name of your file. Same filename will be overwritten in the cloud.]
3. Content-Range: bytes \*/**482** [Replace it by the size in bytes of your desired file]
4. Content-Length: 0 [This need to be 0 on the handshake]

This request will return nothing in the body but the wanted information is in the **ETag** header. You will need to pass it through the next Http Request.

Here’s the second Http Request

**PUT** <https://s3mxcloudservice.cloudapp.net:443/v2.3/Repositories/S3MXECPlugin--Server/S3MX/Document/Production_Graz_3MX~2FScene~2FProduction_Graz_3MX.3mx/$file>

Required headers:

1. Authorization: token xyz [Replace xyz by a valid IMS token]
2. Content-Type: text/plain
3. If-Match: “ETag” [Make sure that this is between quotation marks. This ETag is from the previous request]
4. Content-Range: bytes **0-481**/**482** [0-481 is the data that you are sending. This can be 0-255 but this mean that you’ll need to do another request container the rest of the file like so 256-481. The 482 is the total file size]
5. Content-Length 482 [Containing the total length of the data that you are sending. If at Content-Range you’ve enter 0-255/482 you’ll need to enter 256 as the Content-Length]

The required body/content is the content of the file wether text of binary; no interpretation will be performed:

{"3mxVersion": 1,

"name":"Production\_Graz\_3MX",

"description":"This is the scene description. It should be between quotation marks.",

"logo":"logo.png",

"sceneOptions":[{"navigationMode":"PAN"}],

"layers": [

{

"type":"meshPyramid",

"id":"mesh0",

"name":"Production\_Graz\_3MX",

"description":"This is a description of the model. It should be between quotation marks.",

"SRS": "EPSG:32633",

"SRSOrigin": [533459,5212605,0],

"root": "Data/Production\_Graz\_3MX.3mxb"

}]}