OSLC API

TWCloud adds OSLC support and now exposes model element data following OSLC Architecture Managemenet (AM) (http://open-services.net/ns/am#) vocabulary. Along with core OSLC services provided in TWCloud, this enables smooth integration with other OSLC-compatible tools by linking resources in Linked Data fashion. Here are the key points behind current TWCloud OSLC provider implementation:

- OSLC root services document URI can be found using the following pattern - *http(s)://TWC_IP:PORT/oslc/rootservices*.
- Each of the model elements is exposed using the following URI pattern - *http(s)://TWC_IP:PORT/oslc/am/{projectID}/{elementID}*.
  *Note: the *elementID* here is an Element Server ID.
- We expose the following model element properties:
  - rdf:type (http://www.w3.org/1999/02/22-rdf-syntax-ns#type) - represents architecture resource type, which according to OSLC AM vocabulary is always a Resource (http://open-services.net/ns/am#Resource).
  - dcterms:modified (http://purl.org/dc/terms/modified) - stands for the last element modification date.
  - dcterms:identifier (http://purl.org/dc/terms/identifier) - the Element Server ID as used in element's URI pattern.
  - dcterms:title (http://purl.org/dc/terms/title) - the name of the element

Sample RDF/XML representation of element data

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:oslc="http://open-services.net/ns/core#"
        xmlns:oslc_am="http://open-services.net/ns/am#"
          xmlns:dc="http://purl.org/dc/terms/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdf:Description rdf:about="https://localhost:8111/oslc/am/b1acff2d-4396-4314-9dba-477d573ede16
/50775427-bce2-4de4-b747-8ab82d294237">
    <rdf:type rdf:resource="http://open-services.net/ns/am#Resource"/>
    <oslc:serviceProvider rdf:resource="https://localhost:8111/oslc/am/b1acff2d-4396-4314-9dba-
477d573ede16/services"/>
    <dcterms:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">May 16, 2018 2:08:52 PM<
/dcterms:modified>
    <dcterms:identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#string">50775427-bce2-4de4-b747-
8ab82d294237</dcterms:identifier>
    <dcterms:title rdf:parseType="Literal">Engine</dcterms:title>
  </rdf:Description>
</rdf:RDF>
```

- Currently we have neither OSLC delegated dialog nor querying services exposed.
- We offer OSLC UI previews through integration with CC4TWC. For more information, check Publishing an OSLC resource.

OAuth 1.0a authentication

In order to ensure secure access to server resources via OSLC, OAuth 1.0a authentication protocol is used.

OAuth 1.0a requires consumer key and secret to be known before starting the authentication process flow. These can be created through TWCloud Admin as described in OAuth consumer keys management.

Alternatively, a pair of consumer key and secret can be generated manually via a service exposed in the root services document. The following HTTP POST request should be made to a consumer key generation service (jfs:oauthRequestConsumerKeyUrl):

HTTP request body

```
{
  "name": "consumerNameGoesHere",
  "secret": "OAuthSecretGoesHere"
}
```

HTTP response body

```
{
  "key": "generatedConsumerKeyShouldBeHere"
}
```

Note, that keys generated following this approach will still need to be approved by an administrator through TWCloud Admin console.