OSLC PSM Strategy

The SysML v2 Services RFP requires a PIM and two standard PSMs, one for REST API and another of OSLC API.

The purpose of the two standard PSMs was to provide the REST API as a minimal, mostly direct reference implementation of the PIM for validating the specifications and providing a simple, easy to consume PSM using OpenAPI that could easily have language specific APIs generated.

The purpose of the OSLC PSM was to enable interoperability and integration between SysML, lifecycle management and other tools.

SysML v2 services API defines a REST API which has a prototype implementation at http://sysml2-sst.intercax.com:9000/and documented at http://sysml2-sst.intercax.com:9000/docs/.

This API and the reference implementation currently does not expose its resources as OSLC Architecture Management resources, or provide any RDF implementation. Rather it provides instances of Element and Relationship objects that provide type and property information.

The SST Track 5 team may have been under the impression that OSLC put a lot of complex requirements on the PSM that they felt were unnecessary, too platform or use case specific, or too difficult to implement. However, there are only three minimal requirements that the REST PSM would need to implement to be compatible with OSLC and linked-data principles. These requirements do not represent significant changes to the current REST PSM and its reference implementation.

Ideally the OSLC PSM would be a layered extension of the REST PSM in order to provide additional integration capabilities. In order for this to happen, the REST PSM would need to support three minimal requirements:

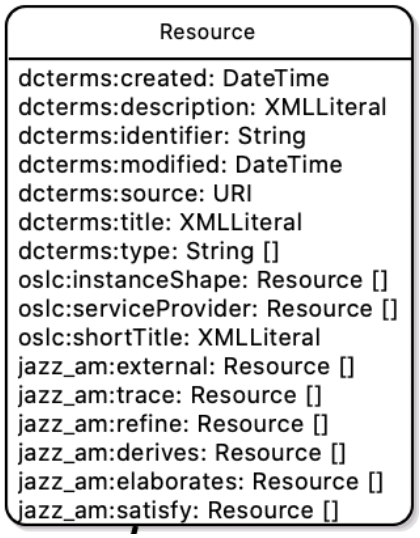
- Element and Relationship subclass oslc_am:Resource

- Support at least one RDF representation, e.g., Accept=application/ld+json
- Support identification of groups of related versioned resources with Configuration-Context header

Here are some of the implications and benefits of these three requirements.

1. Element and Relationship subclass oslc_am:Resource.

oslc_am:Resource has the following standard properties



Of these, only the following are required (all others have multiplicity 0..* and are therefore optional):

- dcterms:identifier
- dcterms:title

Aligning Element and Relationship to oslc_am:Resource would require trivial renames or (JSON-LD aliases) in Element and Relationship properties - a small price to pay for interoperability and integration.

2. REST PSM requires Accept=application/ld+json

SysML models can be quite complex, are often developed across organization and ownership boundaries, are reusable, etc., characteristics of many WWW vocabularies. It's critical that even the minimal REST PSM support scalability at the WWM level. JSON alone

does not do that. To support realistic use cases, the REST PSM should require JSON-LD. This is a small price to pay for open-world assumptions, RDF and OSLC compatibility, and open extensible integration.

The REST API already supports JSON, extending this to JSON+LD would simply provide additional information vocabulary and type information for search engines and reflective tool development.

This would have the following benefits, while maintaining precision and simplicity of JSON used in the REST PSM:

- Integrates the SysML and RDF worlds through a common resource representation, a huge opportunity for expanding the use of SysML
- @Context provides additional metadata information for search engines and reflective tools
- @id and URLs to scale JSON to support linked data to the whole WWW
- Provides namespace and identity capabilities to JSON data to avoid ambiguity or name collisions

For example, here are to equivalent RDF representations of a SysML Block (as a sublcass of oslc_am:Resource), one for Turtle and another for JSON-LD

```
@prefix oslc_am: <http://open-services.net/ns/am#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix oslc:  <http://open-services.net/ns/core#> .
@prefix sysml: <http://omg.org/ns/sysml#> .

<https://acme.com/resources/
e7b4a34d-26b5-434c-8bb1-1e8224e7eb8e>
      a                  sysml:Block ;
      oslc:serviceProvider  oslc_am: ;
      dcterms:identifier    "e7b4a34d-26b5-434c-8bb1-1e8224e7eb8e" ;
      dcterms:title         "VehicleA" .

{
```

```
  "@context" : "http://omg.org/contexts/sysml",
  "@id" : "https://acme.com/resources/
e7b4a34d-26b5-434c-8bb1-1e8224e7eb8e",
  "@type" : "sysml:Block",
  "identifier" : "e7b4a34d-26b5-434c-8bb1-1e8224e7eb8e",
  "title" : "VehicleA",
  "serviceProvider" : "http://open-services.net/ns/am#"
}
```

The JSON-LD example assumes that http://omg.org/context/sysml provides the JSON-LD schema for SysML, something that can be generated from the SysML v2 metamodel. As you can see, the JSON-LD representation of an RDF and OSLC architecture management resource doesn't look very different than the REST PSM Element JSON.

3. Support optional Configuration-Context header

Adding to the characteristics of SysML model above are the need to support versions and variants of integrated models. Doing this in a scalable way across organization, ownership, and repository boundaries will require the notion of contributions from multiple sources. Since the REST PSM does specify versioning capability, it should also support a means of identifying consistent versions of related resources that can be used to establish a context for reuse. A Configuration-Context header with a URL value to identify that context (e.g., commit, change-set, stream, baseline, global configuration, etc.) would establish a standard way of expressing organizations of versioned resources.

OSLC Prototype Implementation

A prototype implementation of the OSLC PSM could be a full featured OSLC server and Architecture Management provider in order to demonstrate linking of SysML resources with other lifecycle resources supported and managed by other tools such as requirements, test cases, change requests, simulations, etc.

The SysML OSLC PSM will define a binding with the PIM that can be implemented by building an OSLC server that exposes the existing REST

PSM as OSLC resources. This keeps the OSLC and REST PSMs separate, while showing how they are related. The adapter can be implemented whether the REST PSM adopts the above requirements or not. Adopting those requirements just simplifies the mapping and enables direct interoperability of tools that implement either PSM.

Lyo Designer can be used to generate a SysML v2 server that as an adapter on the existing REST PSM prototype implementation. This would provide at least a minimal UI, and would demonstrate integration with other OSLC servers.

SysML Vocabulary Instances and transformations

The OSLC server will support some vocabulary for SysML. That vocabulary could be the SysML v2 Services API PIM model - focusing on Element and Relationship, or it could expose the whole SysML metamodel as first-class RDF resources.

Supporting the SysML Services API PIM model would be relatively easy to define and implement using Lyo Designer. The domain specification (vocabulary and resource shapes) for the PIM model is reasonably simple and can be implemented by hand. An OSLC server supporting this domain would provide RDF representations of the PIM elements, similar to what the REST PSM is providing.

To support the full SysML metamodel, an OSLC domain specification consisting of vocabulary terms and constraining resource shapes would need to be created. We belive this can be automatically generated from the SysML v2 metamodel

Then Lyo Designer can be used to create an OSLC server for that OSLC SysML domain specification, and that server can also generate Swagger.io documentation of the OSLC REST API, and generate the required JSON-LD schema.