Open Services for Lifecycle Collaboration
Core Specification Version 2.0
Appendix B: Representations and Examples

This Version
- http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixRepresentations

Latest Version
- http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixRepresentations

Authors
- Dave Johnson

Contributors
- The OSLC Core Specification Workgroup

Table of Contents

- Guidelines for application/rdf+xml
  - RDF/XML Examples
- Guidelines for application/xml
  - Steps to generate an XML representation
  - XML Examples
- Guidelines for JSON
  - JSON Examples
- Turtle Representations
  - Turtle Examples
- Atom Representations
  - Steps for forming an Atom format OSLC Defined Resource representation
  - Steps for forming an Atom format OSLC Query Resource representation
  - Atom Examples

This document provides details on how to form and serve RDF/XML, XML, Turtle, JSON and Atom representations of OSLC defined resources.
Recall from the OSLC Core specification section on OSLC Defined Resource Representations that "OSLC services SHOULD provide and accept RDF/XML representations for each OSLC resource." An RDF/XML representation SHOULD be:

- Returned when these content types requested: application/rdf+xml
- Returned as content-type: application/rdf+xml

OSLC resources are defined using the RDF data model and so, the first guidance for both parsing and generating RDF/XML representations is to use an RDF/XML toolkit. Follow the RDF/XML specifications and your toolkit's documentation. There are such toolkits available for most popular languages.

RDF/XML Examples

See separate page OSLC Core Spec RDF XML Examples

Guidelines for application/xml

In this section we offer instructions for creating an XML representation for OSLC Defined Resources. The resultant XML representations are designed to validate as RDF/XML and therefore are often referred to as "constrained RDF/XML" or "abbreviated RDF/XML" representations, however the main intent of these instructions is to document a specific XML representation format for use in scenarios where a more unfettered RDF/XML representation or use of RDF/XML toolkits is not appropriate or preferred. OSLC domain specifications SHOULD refer to these instructions when requiring application/xml representations.

XML representations returned by OSLC Defined Resources SHOULD be:
- Returned when these content types requested: application/xml
- Returned as content-type: application/xml

Steps to generate an XML representation

This set of steps MAY be followed to form an XML representation of an OSLC Defined Resource. Other techniques MAY be used as long as the resulting XML is the same.

1.0 Open the XML root element <rdf:RDF> and include all required namespace declarations
2.0 Open the XML element as a first child of the XML root element using the Resource's Prefixed Name
   2.1 Add to the XML element the rdf:about attribute set to the URI of the resource
3.0 Represent the properties of the resource (using the property rules below)
4.0 Close the XML element (using Resource's Prefixed Name)
5.0 If this resource is a partial resource returned as a result of Resource Paging then:
   5.1 Open the XML element <oslc:ResponseInfo>
   5.2 Represent the properties of the oslc:ResponseInfo resource (using the property rules below)
      5.2.1 If this is not the last page, the oslc:nextPage property-value MUST point to the next page.
   5.3 Close the XML element </oslc:ResponseInfo>
These are the property rules referred to above. OSLC property values are represented as XML elements with QNames formed using the Property's QName namespace prefix and name.

1. For each property value of the resource
   - 1.1 Open XML element using QName of property
   - 1.2 If property value is a Literal value-type
     - 1.2.1 Inside the XML element add the value as a string with any required escaping
   - 1.3 If property value is value-type Resource with representation of Reference
     - 1.3.1 Add to the XML element the rdf:resource attribute set to URI
     - 1.3.2 If the resource reference is an Anchor (see OslcCoreSpecAppendixLinks Guidance) then add rdf:ID with an ID, unique to the resource and that will be referenced from a separate annotations element (see below).
   - 1.4 If property value is value type Local Resource with representation of Reference
     - 1.4.1 Add to the XML element the rdf:nodeID attribute set to local ID of the resource referred to
   - 1.5 If property value is value-type Resource with representation of Inline
     - 1.5.1 Open XML element using QName of the resource being represented inline
     - 1.5.2 Add to the XML element the rdf:about attribute set to URI of resource
     - 1.5.3 Inside the XML element represent property values of the linked resource
     - 1.5.4 Close the XML element
   - 1.6 If property value is value-type Local Resource with representation of Inline
     - 1.6.1 Open XML element using QName of the resource
     - 1.6.1 If this Local Resource is to be referred to elsewhere within the (containing) resource representation, then add to the XML element to rdf:nodeID set to the local ID you wish to assign to the resource.
     - 1.6.3 Inside the XML element represent property values of the resource
     - 1.6.4 Close the XML element
   - 1.7 Close XML element

If there are any Anchors (see step 1.3.2 above) then for each:

- 1.0 Open XML rdf:Description element with an rdf:about that refers to the anchor's ID.
  - 1.1 Inside the XML element represent each property value of the anchor, following the property rules above.
- 2.0 Close the XML rdf:Description element

XML Examples

The examples on the OSLC Core Spec RDF XML Examples were generated using the rules above.

Guidelines for JSON

An OSLC Service MAY provide and understand JSON representations of OSLC Defined Resources and these representations SHOULD:
The first round of OSLC specifications defined a simple JSON format for representing OSLC resources and until there is a standard JSON representation for RDF data, we expect that OSLC domain specifications will continue to use this simple format.

The Core workgroup has some minor changes to the format to allow prefixes to be declared and offers guidance in the form of step-by-step instructions on how to generate it.

This set of steps MAY be followed to form a JSON representation of an OSLC Defined Resource. Other techniques MAY be used as long as the resulting JSON is the same.

1. Create a JSON object with the following fields:
   1.1 Set `rdf:about` to the URI of the resource.
   1.2 Set `rdf:type` to the resource's type.
   1.3 Set `prefixes` to a JSON object with one field per prefix used in the file, the field value should be the prefix base that corresponds to the prefix.

2. Add to the JSON object fields to represent the property values of the resource (using the property rules below).

3. If this resource is a partial resource returned as a result of Resource Paging then:
   3.1 Add to the JSON object a field `oslc:responseInfo` with value-type `oslc:ResponseInfo`
      3.1.1 Represent the property values of the response info resource (using the property rules below)
      3.1.2 If this is not the last page, the `oslc:nextPage` property-value MUST point to the next page.

These are the property rules referred to above. For each property:

1. For each property of a resource
   1.1 If the property is specified as a single-valued property, then make field's value literal or JSON Object
   1.2 If value-type is `Literal`
      1.2.1 If value-type is a Literal String, `DateTime`?, or `XMLLiteral`, set the field's value to a JSON string representation of the property value.
      1.2.2 If value-type is a Literal numeric (Integer, Decimal, Double, Float), set the field's value to a JSON number.
      1.2.3 If value-type is a Literal boolean, set the field's value to a JSON boolean.
   1.3 Else if value-type is `Resource` with representation of `Reference`
      1.3.1 Set field's value to JSON object with field:
         1.3.1.1 Set `rdf:resource` to URI of resource
      1.3.2 If the resource reference is part of an Anchor (as described in Link Guidance) and has annotating property values, then include these properties within the JSON object.
1.4 Else if value-type is **Local Resource** with representation of **Reference**
   - 1.4.1 Set field's value to JSON object with field:
     - `rdf:nodeID` set to local ID of resource

1.5 Else if value-type is **Resource** with representation of **Inline**
   - 1.5.1 Set field's value to JSON object with fields:
     - `rdf:about` set to URI of resource whose values are being inlined
     - Represent properties of inlined resource (using these same rules)

1.6 Else if value-type is **Local Resource** with representation of **Inline**
   - 1.6.1 Set field's value to JSON object with fields:
     - `rdf:nodeID` set to local ID you wish to assign to this resource. Only needed when Local Resource needs to refer to this resource.
     - Represent properties of inlined resource (using these same rules)

### JSON Examples

See separate page [OSLC Core Spec JSON Examples](#)

### Turtle Representations

An OSLC Service **MAY** provide and understand Turtle representations of OSLC Defined Resources.

- Returned when these content types requested: `application/x-turtle`
- Returned as content-type: `application/x-turtle`

As with RDF/XML we recommend using an RDF toolkit to parse and generate Turtle. Turtle is simple enough that we do not need to offer step-by-step guidance on how to generate it.

### Turtle Examples

See separate page [OSLC Core Spec Turtle Examples](#)

### Atom Representations

An OSLC Service **MAY** provide and accept Atom Syndication Format representations of OSLC Defined Resources (reference: Atom Format).

#### Steps for forming an Atom format OSLC Defined Resource representation

Note that there are special steps below for creating an OSLC Query Resource representation in Atom format.

When a client requests an Atom representation of an OSLC Defined Resource and specifies via HTTP Accept header content-type `application/atom+xml` or `application/atom+xml;type=entry` then an OSLC Service **SHOULD** respond with an Atom representation.

To represent an OSLC Defined Resource in Atom format we use an Atom `<entry>` element as document root, we
map some OSLC properties to Atom elements and inside the Atom <content> element, we place an RDF/XML representation (as defined above) of the resource that we are representing. Atom representations **MUST** comply with the rules of Atom Syndication Format (reference: Atom format).

- 1.0 Open an Atom <entry> element
  - 1.1 Add Atom <id> element. Atom requires a unique ID here and in the form of a URN, the resource URI **MAY** be used here.
  - 1.2 Add Atom <title> element. Atom requires a non-empty title value. A `dcterms:title` value **MAY** be used here.
  - 1.3 Add Atom <updated> element. Atom requires an updated date-time value. In cases where an updated-time value is difficult or expensive to compute, OSLC domain specifications might wish to declare this value to be insignificant.
  - 1.4 Add Atom <content> element. Atom representations **SHOULD** provide a content element that contains an RDF/XML representation of the resource being represented.
  - 1.5 Add Atom <author> element. Atom requires an author element.

- 2.0 Close Atom <entry> element

**Steps for forming an Atom format OSLC Query Resource representation**

When a client requests an Atom representation of an OSLC query resource and specifies via HTTP Accept header content-type `application/atom+xml` or then an OSLC Service **SHOULD** respond with an Atom feed representation as described below.

This representation guidance below represents each query result within the Atom feed as an Atom <entry> with RDF/XML data inlined inside the Atom entry's <content> element. Unless otherwise specified by the relevant OSLC domain specification, clients **SHOULD** expect that this inlined content will be full and unconstrained RDF/XML.

To represent an OSLC Query Resource in Atom format we use an Atom <feed> element as document root, add the required properties and then represent each member property value returned in the query response as an Atom <entry> element.

- 1.0 Open an Atom <feed> element
  - 1.1 To that element add as children the required Atom elements:
    - atom:id Must be a unique URN. The query resource URI may be used.
    - atom:title With same value as the query's `oslc:ResponseInfo` value for `dcterms:title`
    - atom:updated Set to time that query result was generated
    - atom:author This element is not meaningful but Atom requires it, so provide only a dummy value.
  - 1.2. To that element add optional Atom elements that you wish to provide:
    - atom:summary Set to `oslc:ResponseInfo` value for `dcterms:description`
    - atom:link With rel="self" set to same value as the query's resource URI
    - atom:link With rel="next" set to same value as the query's `oslc:ResponseInfo` value for `oslc:nextPage`
  - 1.3 For each member property value returned by the query:
1.3.1 Open an Atom <entry> element

1.3.2 Add to that element the required Atom elements:

- **atom:id** Must be a unique URN. The member value's URI may be used.
- **atom:title** If the member value has a dcterms:title value, then use that. Otherwise provide a non-empty string with only a dummy value.
- **atom:updated** If the member value has a dcterms:modified value then use that. Otherwise use same value used at feed level.
- **atom:author** If the member value provides dcterms:creator values, then each should be represented as an atom:author element.

1.3.3 Add to the that element the atom:content element.

- Set the atom:content element's src attribute to the URI of the member value
- Set the atom:content element's type attribute to application/rdf+xml
- Inside the content element, include a complete RDF/XML representation of the member value as specified by the query, i.e. may be a subset of the resource's values as specified by the query syntax string. Set the rdf:about attribute of the root element to the URI of the query result.

1.3.4 Close the Atom </entry> element

2.0 Close the Atom </feed> element

## Atom Examples

See separate page [OSLC Core Spec Atom Examples](#)

---

Topic revision: r18 - 28 Oct 2011 - 16:07:20 - [MichaelFiedler](#)

Main.OSLCCoreSpecAppendixRepresentations moved from Main.OSLCCoreRepresentationsGuidance on 22 Oct 2010 - 18:37 by [DaveJohnson](#) - put it back