OSLC Core Version 3.0. Part 5: Attachments

Committee Specification 01

04 April 2017

Specification URIs

This version:
http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/cs01/part5-attachments/oslc-core-v3.0-cs01-part5-attachments.html
http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/cs01/part5-attachments/oslc-core-v3.0-cs01-part5-attachments.pdf

Previous version:
http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd01/part5-attachments/oslc-core-v3.0-csprd01-part5-attachments.html
http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd01/part5-attachments/oslc-core-v3.0-csprd01-part5-attachments.pdf

Latest version:
http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/oslc-core-v3.0-part5-attachments.html

Technical Committee:
OASIS OSLC Lifecycle Integration Core (OSLC Core) TC

Chairs:
Jim Amsden (jamsden@us.ibm.com), IBM
Martin Sarabura (msarabura@ptc.com), PTC

Editor:
Jim Amsden (jamsden@us.ibm.com), IBM

Additional artifacts:
This specification is one component of a Work Product that also includes:

Related work:
This specification is related to:

RDF Namespaces:
http://open-services.net/ns/core#

Abstract:
Binary or text documents may be considered attachments to other resources. This specification describes a minimal way to manage attachments related to web resources using LDP-Containers and Non-RDF Source [LDP].

Status:
This document was last revised or approved by the OASIS OSLC Lifecycle Integration Core (OSLC Core) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=oslc-core#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC’s public comment list oslc-core-comments@lists.oasis-open.org, after subscribing to it by following the instructions at the “Send A Comment” button on the TC’s web page at https://www.oasis-open.org/committees/oslc-core/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC’s web page (https://www.oasis-open.org/committees/oslc-core/ipr.php).

Note that any machine-readable content (aka Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:
When referencing this specification the following citation format should be used:

[OSLC-Attachments-3.0]

Notices
Copyright © OASIS Open 2017. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL
1. Introduction

*This section is non-normative.*

Various tools handle the association and creation of related resources in conceptually similar ways, but often differ in details on how it is accomplished. The Linked Data Platform (LDP) already defines a model by which it is possible to relate resources to another, even if they are not RDF-based. This specification defines the method to create associated attachments to a given resource and understand if that resource supports the attaching of attachments.

As an example of how to create an attachment, simply HTTP POST the attachment content to the attachment container for the
A LDP-NR whose lifecycle is coupled with the attaching resource.

**Attachment Container**

A LDPC that contains Attachments for a resource.

**Attachment Descriptor**

A LDP-RS that contains additional data about an Attachment.

### 1.2 References

#### 1.2.1 Normative references

**[HTTP11]**


**[LDP]**

Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

**[OSLCCore3]**


**[RFC2119]**


**[RFC2183]**


**[RFC5023]**

1.3 Typographical Conventions and Use of RFC Terms

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words MUST, MUST NOT, REQUIRED, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this specification are to be interpreted as described in [RFC2119].

The namespace for OSLC Core is http://open-services.net/ns/core#.

Sample resource representations are provided in text/turtle format [turtle].

Commonly used namespace prefixes:

```
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix ldp:     <http://www.w3.org/ns/ldp#>.
@prefix oslc:    <http://open-services.net/ns/core#>.
@prefix oslc_cm: <http://open-services.net/ns/cm#>.
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix wdrs:    <http://www.w3.org/2007/05/powder-s#>. 
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#>.
```

2. Motivation

This section is non-normative.

Most users of lifecycle tools have the need to easily create attachments across a variety of integrated tools and associate them to some lifecycle resource in context to some scenario. Some specific scenarios where this touches cross tool integration:

- **Running application scanning**: automatically creating a defect or task to track a problem, attaching a log file that outlines the details of the problem.
- **Publishing build results**: as part of an automated software build, publish successful build artifacts to an asset management repository
- **Mockups of app design** : share screenshots and designs to a given user story (requirement)

3. Basic Concepts

This section is non-normative.

Attachments are added to a resource via a simple POST request to the appropriate LDP-Container resource. The entity body becomes the content of the attachment resource. The attachment may automatically be associated with the resource via some membership relationship, which may use the oslc:attachment membership predicate. Statements are also automatically added to the oslc:AttachmentDescriptor resource. The property values are assigned by the server or can be determined from standard headers of the POST. The following table maps the HTTP request headers from the POST request to create the attachment resource, to what can be used to derive the initial values in the indicated oslc:AttachmentDescriptor resource:

<table>
<thead>
<tr>
<th>HTTP Request Header</th>
<th>Prefixed Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slug</td>
<td>dcterms:title</td>
</tr>
<tr>
<td>Content-Type</td>
<td>dcterms:format</td>
</tr>
<tr>
<td>Content-Length</td>
<td>oslc:attachmentSize</td>
</tr>
</tbody>
</table>
4. Working with Attachments

This section is non-normative.

The following examples illustrate how a client can work with attachments.

4.1 Find the Attachments for a Resource

Clients get the attachments for a resource by:

1. Finding the attachment container for a resource using an HTTP OPTIONS method and Link header
2. Getting the container for the list of attachments

Each resource that supports attachments has an attachment container, which is an LDP container. Clients discover the attachment container through an HTTP Link header. A client can use GET or HEAD to get the Link header, but OPTIONS is often more efficient because the server does not have to calculate the ETag or content length of the response. LDP resources must support HTTP OPTIONS, and responses to all HTTP requests for resources that support attachments must have the Link header.

```plaintext
EXAMPLE 3
OPTIONS /bugs/2314 HTTP/1.1
Host: example.com

The response contains a Link to the attachment container with Link relation http://open-services.net/ns/core#AttachmentContainer. Note that other Link headers are in the response. In fact, LDP requires additional Link headers, which is why the response has a Link with relation type and target URI http://www.w3.org/ns/ldp#Resource.

EXAMPLE 4
HTTP/1.1 200 OK
Allow: GET,HEAD,OPTIONS,PUT,DELETE
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type",
     <http://example.com/bugs/2314/attachments>; rel="http://open-services.net/ns/core#AttachmentContainer"

Now the client requests the attachment container to see the attachments for this resource. It's a good practice to include an HTTP Prefer header to explicitly ask the server for the LDP containment triples.

EXAMPLE 5
GET /bugs/2314/attachments HTTP/1.1
Host: example.com
Accept: text/turtle
Prefer: return=representation; include="http://www.w3.org/ns/ldp#PreferContainment"

The response is an LDP container for the attachments. It can be any LDP container such as an ldp:BasicContainer or an ldp:DirectContainer. This example uses an ldp:BasicContainer. The attachment container only contains attachments for a single resource.

EXAMPLE 6
HTTP/1.1 200 OK
Allow: GET,HEAD,OPTIONS,POST
Content-Length: 323
Content-Type: text/turtle
ETag: W/"2773fef2237e91273bde782a43925458"
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type",
     <http://www.w3.org/ns/ldp#Container>; rel="type",
     Preference-Applied: return=representation
Vary: Accept,Prefer

@prefix oslc: <http://open-services.net/ns/core#> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .

<http://example.com/bugs/2314/attachments>
    oslc:AttachmentContainer , ldp:BasicContainer .
    ldp:contains <http://example.com/bugs/2314/attachments/2> ,
    <http://example.com/bugs/2314/attachments/1> .

Clients can look at the ldp:contains property on the container for the attachments.

4.2 Get the Attachment Content
Once clients have the attachment URI, they can get the attachment by simply making an HTTP GET request to the attachment URI.

A **Slug** header can be included by a server in the response to a GET on an attachment resource. If a client wishes to store the content as a file, this value provides a hint as to the file name to use (subject, of course, to any file system restrictions). In the absence of a **Slug** header, the client may use the last segment of the resource's URI as a hint, or just choose an arbitrary file name.

**EXAMPLE 7**

```
GET /bugs/2314/attachments/1 HTTP/1.1
Host: example.com
```

The response body is the attachment content. Servers should set the response **Content-Type** to describe the media type of the attachment. The response may have a **Content-Disposition** header with a filename parameter, although this isn't required. This example also contains a Link with relation **describedby**, which links to the **oslc:AttachmentDescriptor** for the attachment.

**EXAMPLE 8**

```
HTTP/1.1 200 OK
Allow: GET,HEAD,OPTIONS,PUT,DELETE
Content-Disposition: attachment; filename="screenshot.png"
Content-Length: 53622
Content-Type: image/png
ETag: W/"678609dceede86e0f8b8aa5f252b84af611"
Link: <http://example.com/bugs/2314/attachments/meta/1>; rel="describedby",
     <http://www.w3.org/ns/idp#Resource>; rel="type",
     <http://www.w3.org/ns/idp#NonRDFSource>; rel="type"
[binary content]
```

**4.3 Create an Attachment**

To create an attachment, POST the attachment content to the attachment container for the resource. The request should have a **Content-Type** header describing the attachment's media type and subtype as specified in Media Types. The optional **Slug** header is used to give the attachment a name. The **Content-Length** header is used to initialize the attachment size.

A client can set a **Slug** header in the attachment-creating POST to specify a hint for a name for the resource as part of that single request. This can be important as some systems require a name at the time the attachment is created. Different systems may have different requirements for valid attachment names, so the value of the **Slug** header should be interpreted as a hint in this context. If the given name can not be used as specified, it is transformed into a valid name. If that is not possible or the header is not specified, an arbitrary value is assigned. Failure due to an invalid name is undesirable because of the potentially large size of an attachment resource.

The client can provide the attachment size in the **Content-Length** header and this can be used to initialize the **oslc:AttachmentDescriptor oslc:attachmentSize** property. The server may compute a different attachment size than that provided by the client if the client specified value is incorrect or not provided.

**EXAMPLE 9**

```
POST /bugs/2314/attachments HTTP/1.1
Slug: design
Content-Type: application/vnd.oasis.opendocument.text
Content-Length: 18124
[binary content]
```

The response contains a Link to the new attachment in the **Location** header. This server has also included a Link to the **oslc:AttachmentDescriptor** for the attachment in the HTTP response, which contains metadata about the attachment.

When a server successfully creates an attachment resource, it responds with an HTTP status code of 201 (created) with the URI of the newly created attachment resource in the HTTP response **Location** header. Additionally, if the server created an associated **oslc:AttachmentDescriptor** resource, the URI for this resource should be listed in the HTTP response **Link** header [RFC5988] with rel="describedby" [LDP].

Properties for the AttachmentDescriptor that are not readOnly, such as its title and description, can be updated using the usual HTTP **PUT** method.

**EXAMPLE 10**

```
HTTP/1.1 201 Created
Allow: GET,HEAD,OPTIONS,POST
```

---

oslc-core-v3.0-c001-part5-attachments

Copyright © OASIS Open 2017. All Rights Reserved.

Page 7 of 11

04 April 2017
4.4 Update an Attachment

To update an attachment, PUT the attachment content to the attachment resource.

**EXAMPLE 11**

PUT /bugs/2314/attachments/3 HTTP/1.1
Content-Type: application/vnd.oasis.opendocument.text
Content-Length: 19377

[binary content]

The server typically responds with a 204 No Content status if the request succeeds. It also updates an associated attachment metadata in the oslc:AttachmentDescriptor in the describedby link. For example, the client could have included a \(\text{Slug} \) header on the update request in order to rename the attachment.

**EXAMPLE 12**

HTTP/1.1 204 No Content
Link: <http://example.com/bugs/2314/attachments/meta/3>; rel="describedby";
anchor="http://example.com/bugs/2314/attachments/3",
<http://www.w3.org/ns/idp#Resource>; rel="type"
Content-Length: 0

4.5 Remove an Attachment

To remove an attachment, make a DELETE request on the attachment URI. This removes the attachment from the container and deletes the content and attachment metadata from the server.

**EXAMPLE 13**

DELETE /bugs/2314/attachments/3 HTTP/1.1
Host: example.com

The server typically responds with 204 No Content status if the request was successful.

**EXAMPLE 14**

HTTP/1.1 204 No Content
Content-Length: 0

4.6 Include Attachment Information Inline with a Resource

Servers can choose to include the attachment information directly in the HTTP response for a resource although this isn’t required. Here is an example defect resource that contains attachments. The attachment container is an \(\text{ldp:DirectContainer} \) where the membership resource is the defect itself. The membership predicate is \(\text{oslc:attachment} \), although this predicate is not required. The following example shows the results of an HTTP GET on \(\text{http://example.com/bugs/2314} \).

**EXAMPLE 15**

```
@prefix oslc: <http://open-services.net/ns/core#> .
@prefix oslc_cm: <http://open-services.net/ns/cm#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix ldp: <http://w3.org/ns/ldp#> .
@prefix wdrs: <http://www.w3.org/2007/05/powder-s#> .

<http://example.com/bugs/2314> a oslc_cm:Defect ;
  oslc:attachment <http://example.com/bugs/2314/attachments/2> ;
  dcterms:title "A serious bug!" ;
  dcterms:identifier "2314" .

<http://example.com/bugs/2314/attachments>
  ldp:DirectContainer , oslc:AttachmentContainer ;
  ldp:contains <http://example.com/bugs/2314/attachments/2> ,
  <http://example.com/bugs/2314/attachments/1> ;
  ldp:hasMemberRelation oslc:attachment ;
```
5. Implementation Conformance

5.1 General

5.1.1 Servers that support OSLC attachments **must** be Linked Data Platform 1.0 conformant servers [LDP].

5.2 Resources with Attachments

5.2.1 Each resource that supports attachments **must** have at least one oslc:AttachmentContainer that holds attachments for that resource.

5.2.2 Responses to HTTP requests for resources that support attachments **must** contain at least one Link header [RFC5988] where the context URI is the resource URI, the Link relation is http://open-services.net/ns/core#AttachmentContainer, and the target URI is the URI of an oslc:AttachmentContainer resource.

5.3 Attachments

5.3.1 An attachment **must** be a conformant Linked Data Platform Non-RDF Source (LDP-NR).

5.3.2 Successful responses to HTTP GET requests for an attachment URI **should** include a Content-Disposition header [RFC2183] with disposition type attachment and a filename parameter. The filename is often the Slug header value used to create the attachment with an appropriate file extension added for the attachment's media type.

5.3.3 If an attachment has an associated oslc:AttachmentDescriptor, responses to HTTP requests for the attachment URI **must** include a Link header [RFC5988] where the context URI is the attachment URI, the Link relation is describedby, and the target URI is the URI of the oslc:AttachmentDescriptor.

5.3.4 When servers update an attachment, they **must** also update any affected oslc:AttachmentDescriptor properties of the associated attachment descriptor.

5.3.5 When deleting attachments, servers **must** also delete any associated oslc:AttachmentDescriptor resources.

5.4 Attachment Containers

5.4.1 Each oslc:AttachmentContainer **must** be a conformant Linked Data Platform Container (LDPC).

5.4.2 Clients **may** use the HTTP Slug request header [RFC5023] to suggest a name when creating an attachment. If present, the Slug header **should not** include a file extension.

5.4.3 Servers **should not** reject an HTTP POST request to an oslc:AttachmentContainer solely because it does not contain a Slug header.

5.4.4 Servers **should not** reject an HTTP POST request to an oslc:AttachmentContainer solely because they cannot use the Slug value unchanged. Servers **should** instead modify the Slug value as needed or assign a different name.

5.4.5 In response to a successful HTTP POST request that creates an attachment with an associated oslc:AttachmentDescriptor, the server **must** include an HTTP Link header in the response where the context URI is the newly-created oslc:AttachmentDescriptor resource URI.
created attachment URI, the link relation is describedBy, and the link target is the oslc:AttachmentDescriptor URI.

5.4.6 Clients MAY specify an LDP-NR interaction model when POSTing RDF content to an oslc:AttachmentContainer by including an HTTP Link header where the target URI is http://www.w3.org/ns/ldp#NonRDFSource and the link relation is type. In this case, Servers MUST honor the client’s requested interaction model and treat the resource as an LDP-NR.

5.4.7 Servers MUST reject an HTTP DELETE request to an oslc:AttachmentContainer.

5.5 Attachment Descriptors

5.5.1 Servers MAY create an associated oslc:AttachmentDescriptor to describe properties of the attachment such as its name, media type, and size.

5.5.2 An oslc:AttachmentDescriptor MUST have an explicit rdf:type Set to http://open-services.net/ns/core#AttachmentDescriptor in its RDF representations. It MAY have additional rdf:type values.

5.5.3 An oslc:AttachmentDescriptor MUST be a conforming Linked Data Platform RDF Source (LDPR).

5.5.4 The dcterms:title of the oslc:AttachmentDescriptor SHOULD be the value of the client-supplied HTTP Slug header.

5.5.5 Servers SHOULD use the Content-Type header value from an attachment creation request to determine the dcterms:format property value in the newly-created attachment’s oslc:AttachmentDescriptor. The dcterms:format value SHOULD be a PURL media-type resource.

5.5.6 An oslc:AttachmentDescriptor MUST conform to the shape defined in 6.1 Resource: AttachmentDescriptor.

6. Resource Constraints

6.1 Resource: AttachmentDescriptor

This document applies the following constraints to the [OSLCCoreVocab] vocabulary terms.

The oslc:AttachmentDescriptor resource type is used to describe the binary resource (or non-RDF Resource) associated with a particular resource. When a client POSTs an attachment content to a server, the server stores the attachment content and assigns a URI just like any other type of resource creation but it may also create an oslc:AttachmentDescriptor resource to contain data about the attachment.

There is no restriction on the content of each attachment resource. For example, it could be a photo of a kitten, an installation manual, a log file, or a source code patch. Since the attachment cannot be expected to contain additional client or server supplied data, a typical set of properties for each attachment is included with the oslc:AttachmentDescriptor resource itself. Thus, the object of each oslc:attachment statement is the binary attachment. Issuing an HTTP HEAD or GET operation on that binary attachment resource URL should produce an HTTP response with a header value of Link: rel='describedBy' to indicate the URL of the oslc:AttachmentDescriptor resource. The properties for the oslc:AttachmentDescriptor resource are indicated in the table below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type URI</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>oslc:AttachmentDescriptor</td>
<td>LDPR-BS to contain data about a LDP-NR(Attachment)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prefix/Name</th>
<th>Occurs</th>
<th>Read-only</th>
<th>Value-type</th>
<th>Representation</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcterms:created</td>
<td>Zero-or-one</td>
<td>true</td>
<td>dateTime</td>
<td>N/A</td>
<td>Unspecified</td>
<td>Timestamp of attachment creation.</td>
</tr>
<tr>
<td>dcterms:creator</td>
<td>Zero-or-many</td>
<td>true</td>
<td>AnyResource</td>
<td>Either</td>
<td>Unspecified</td>
<td>Creator or creators of the attachment. Likely a foaf:Person, but not necessarily so.</td>
</tr>
</tbody>
</table>

*Copyright © OASIS Open 2017. All Rights Reserved. 04 April 2017 Page 10 of 11*
Appendix A. Change History

This section is non-normative.

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>04 April 2017</td>
<td>Jim Amsden</td>
<td>CS was approved and published.</td>
</tr>
</tbody>
</table>