OSLC Automation Specification Version 2.1

Want to contribute?

1. Register (/forums/member/register/)
2. Complete Members Agreement (/legal-agreements/members-agreement/)
3. Complete WPA (/legal-agreements/automation-wpa)

Mailing list (/open-services.net/mailman/listinfo/oslc-automation_open-services.net)

Historical information (/open-services.net/workgroups/automation)

Status: 2.1 Finalization Specification - 17 February 2015

This Version


Latest Version


Previous Version


Authors

- Michael Fiedler (/open-services.net/forums/member/265)
- John Arwe (/open-services.net/forums/member/149)
- Charles Rankin (/open-services.net/forums/member/281)
- Paul McMahan (/open-services.net/forums/member/272)
- Martin Pain (/open-services.net/forums/member/443)
- Umberto Caselli (/open-services.net/forums/member/414)

Contributors

- See Contributors section below.

Contents

- Introduction
  - Terminology
- Base Requirements
  - Compliance
    - Requirements on OSLC Consumers
    - Requirements on OSLC Service Providers
  - Specification Versioning
  - Namespaces
  - Resource Formats
  - Authentication
  - Error Responses
  - Pagination
  - Labels for Relationships
- Automation Resource Definitions
  - Resource: AutomationPlan
    - AutomationPlan Properties
  - Resource: AutomationRequest
    - AutomationRequest Properties
  - Resource: AutomationResult
    - AutomationResult Properties
  - Resource: ParameterInstance
    - ParameterInstance Properties
  - Resource: Dialog
- Automation Service Provider Capabilities
  - Asynchronous and Synchronous Automation Execution
    - Automation Provider Sub-Domains
      - Sub-domain Example
  - Resource Shapes
  - Service Provider Resource
OSLC Automation consumers and service providers SHOULD be compliant with both the core specification and this Automation specification, and SHOULD follow all the guidelines and recommendations in both these specifications.

The following table summarizes the requirements from OSLC Core Specification as well as some (but not all) additional requirements specific to Automation. See the full content of the Automation specification for further details on each of these requirements. Any consumer or service provider behaviors are allowed unless explicitly prohibited by this or dependent specifications; conditional permissive requirements, especially those qualified with "MAY", are implicitly covered by the preceding clause. While technically redundant in light of that broad permission, OSLC specifications do still make explicit "MAY"-qualified statements in cases where the editors believe doing so is likely to add clarity.

### Base Requirements
#### Compliance

This specification is based on OSLC Core Specification (http://open-services.net/bin/view/Main/OslcCoreSpecification). OSLC Automation consumers and service providers MUST be compliant with both the core specification and this Automation specification, and SHOULD follow all the guidelines and recommendations in both these specifications.

### Requirements on OSLC Service Providers
#### Requirement | Level | Origin(s) | Meaning
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown properties and content MUST</td>
<td>Core (<a href="http://open-services.net/bin/view/Main/OslcCoreSpecification">http://open-services.net/bin/view/Main/OslcCoreSpecification</a>)</td>
<td>OSLC clients MUST preserve unknown content</td>
<td></td>
</tr>
<tr>
<td>Unknown properties and content SHOULD</td>
<td>Core (<a href="http://open-services.net/bin/view/Main/OslcCoreSpecification">http://open-services.net/bin/view/Main/OslcCoreSpecification</a>)</td>
<td>OSLC clients SHOULD assume an OSLC service will discard unknown property values</td>
<td></td>
</tr>
</tbody>
</table>
When Automation Consumers request:

For HTTP PUT/POST request formats for Automation resources,

For HTTP GET requests on all OSLC Automation and OSLC Core defined resource types,

See [OSLC Core Representations](http://open-services.net/bin/view/Main/OslcCoreSpecification#Guidelines_for_application_rdf_xml).

In addition to the requirements for

In addition to the namespace URIs and namespace prefixes defined in the OSLC Core specification ([http://open-services.net/bin/view/Main/OslcCoreSpecification](http://open-services.net/bin/view/Main/OslcCoreSpecification)), OSLC Automation defines the namespace URI of OSLC Automation:

```
http://open-services.net/bin/view/Main/OslcAutomation
```

This namespace URI and prefix are used to designate the resources defined in this specification and their properties.

### Resource Formats

In addition to the requirements for OSLC Defined Resource Representations ([http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations](http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations)), this section outlines further refinements and restrictions. See [HTTP Method support table](http://open-services.net/bin/view/Main/OslcCoreSpecification#HTTP_Method_support_table) for further clarification on support for HTTP methods and media types for each OSLC Automation resource.

#### For HTTP GET requests on all OSLC Automation and OSLC Core defined resource types,

- **Automation Providers** MUST provide RDF/XML representations. The RDF/XML representation SHOULD follow the guidelines outlined in the OSLC Core Representations Guidance ([http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations](http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations)).
- **Automation Providers** MAY provide XML and JSON representations. If provided, the XML and JSON representations SHOULD follow the guidelines outlined in the OSLC Core Representations Guidance ([http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations](http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations)).
- **Automation Providers** SHOULD support a [X]HTML representation and a user interface (UI) preview as defined by UI Preview Guidance ([http://open-services.net/bin/view/Main/OslcCoreSpecification#UI_Preview_Guidance](http://open-services.net/bin/view/Main/OslcCoreSpecification#UI_Preview_Guidance)).
- **Automation Providers** SHOULD support error responses as outlined in the OSLC Core Representations Guidance ([http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations](http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations)).

For HTTP GET response formats for Query requests,

- **Automation Providers** MUST provide RDF/XML and MAY provide JSON, XML, and Atom Syndication Format XML.

When Automation Consumers request:

- **application/rdf+xml** Automation Providers MUST respond with RDF/XML representation without restrictions.
- **application/xml** Automation Providers SHOULD respond with OSLC-defined abbreviated XML representation as defined in the OSLC Core Representations Guidance ([http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations](http://open-services.net/bin/view/Main/OslcCoreSpecification#OSLC_DEFINED_Resource_Representations)).

### Specifying Versioning

See [OSLC Core Specification Versioning](http://open-services.net/bin/view/Main/OslcCoreSpecification#Versioning).

### Namespaces

In addition to the namespace URIs and namespace prefixes defined in the OSLC Core specification ([http://open-services.net/bin/view/Main/OslcCoreSpecification](http://open-services.net/bin/view/Main/OslcCoreSpecification)), OSLC Automation defines the namespace URI of OSLC Automation with a namespace prefix of `oae_auto`.

```
http://open-services.net/ns/auto#
```

This namespace URI and prefix are used to designate the resources defined in this specification and their properties.
Authentication

See OSLC Core Authentication section [link]. OSLC Automation puts no additional constraints on authentication.

Error Responses

See OSLC Core Error Responses section [link]. OSLC Automation puts no additional constraints on error responses.

Pagination

OSLC Automation service providers **SHOULD** support pagination of query results and **MAY** support pagination of a single resource’s properties as defined by the OSLC Core Specification.

Labels for Relationships

Automation relationships to other resources are represented as properties whose values are the URI of the object or target resource. When an Automation relationship property is to be presented in a user interface, it may be helpful to provide an informative and useful textual label for that relationship instance. (This in addition to the relationship property URI and the object resource URI, which are also candidates for presentation to a user.) To this end, OSLC providers **MAY** support a `dcterms:title` link property in Automation resource representations, using the anchor approach outlined in the OSLC Core Links Guidance [link].

**RDF/XML and XML example using reified statement:**

```xml
<rdf:RDF
    xmlns:dcterms="http://purl.org/dc/terms/
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#
    xmlns:oslc_auto="http://open-services.net/ns/auto#">
    <oslc_auto:AutomationResult rdf:about="http://example.com/results/4321">
        <oslc_auto:reportsOnAutomationPlan rdf:ID="link1"
            rdf:resource="http://example.com/plans/123" />
    </oslc_auto:AutomationResult>
    <rdf:Description rdf:about="#link1">
        <dcterms:title>Build Definition 123: Pet Shop App production build</dcterms:title>
    </rdf:Description>
</rdf:RDF>
```

Automation Resource Definitions

The Automation resource properties are not limited to the ones defined in this specification; service providers may provide additional properties. It is recommended that any additional properties exist in their own unique namespace and not use the namespaces defined in this specification.

A list of properties is defined for each type of resource. Most of these properties are identified in OSLC Core Appendix A: Common Properties [link]. Any exceptions are noted. Relationship properties refer to other resources. These resources may be in any OSLC domain (including Automation).

The diagram below shows the relationships between Automation Resources.

![Automation Resource Diagram](image)

For all resource types defined in this specification, all **required** properties (those defined with an occurrence of **exactly-one** or **one-or-many**) **MUST** exist for each resource and must be provided when requested. All other properties are optional, and might not exist on some or any resources; those that do not exist will not be present in the returned representation even if requested, while those that do exist **MUST** be provided if requested. Providers **SHOULD** define additional provider-specific properties; providers **MAY** use their own namespaces for such properties, or use standard Dublin Core or RDF namespaces and properties where appropriate.

If no specific set of properties is requested, all properties are returned – both those defined in this specification as well as any provider-specific ones. See Selective Property Values [link] in OSLC Core Specification.
Consumers of OSLC Automation services should note that some resources may have a very large number of related resources, and that some resources may be very large and/or expensive to compute. For this reason, consumers are strongly encouraged to use the oslc.properties parameter to limit the properties returned from a request to the subset required. See Selective Property Values (http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#Selective_Property_Values) in OSLC Core Specification.

**Resource: AutomationPlan**

- **Name:** AutomationPlan
- **Description:** A resource representing the unit of automation which is available for execution.
- **Type URI:** http://open-services.net/ns/auto#AutomationPlan

### AutomationPlan Properties

<table>
<thead>
<tr>
<th>Prefixed 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:contributor</td>
<td>zero-or-many</td>
<td></td>
<td>AnyResource</td>
<td>Either</td>
<td>any</td>
<td>contributor of resource (reference: Dublin Core). It is likely that the target resource will be an foaf:Person resource, but that is not necessarily the case.</td>
</tr>
<tr>
<td>dcterms:created</td>
<td>zero-or-one</td>
<td>True</td>
<td>DateTime</td>
<td>n/a</td>
<td>n/a</td>
<td>Timestamp of resource creation (reference: Dublin Core)</td>
</tr>
<tr>
<td>dcterms:creator</td>
<td>zero-or-many</td>
<td></td>
<td>AnyResource</td>
<td>Either</td>
<td>any</td>
<td>creator of resource (reference: Dublin Core). It is likely that the target resource will be an foaf:Person resource, but that is not necessarily the case.</td>
</tr>
<tr>
<td>dcterms:propertyDefinition</td>
<td>zero-or-one</td>
<td>True</td>
<td>Resource</td>
<td>Reference</td>
<td>n/a</td>
<td>propertyDefinition (<a href="http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#PropertyDefinition">http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#PropertyDefinition</a>) serves the purpose of providing hints to a consumer about the nature of a property of the target resource.</td>
</tr>
<tr>
<td>rdf:type</td>
<td>zero-or-many</td>
<td></td>
<td>String</td>
<td>n/a</td>
<td>n/a</td>
<td>type of resource (reference: Dublin Core).</td>
</tr>
<tr>
<td>dcterms:subject</td>
<td>zero-or-many</td>
<td></td>
<td>String</td>
<td>n/a</td>
<td>n/a</td>
<td>subject of resource (reference: Dublin Core).</td>
</tr>
<tr>
<td>dcterms:title</td>
<td>exactly-one</td>
<td>True</td>
<td>Resource</td>
<td>Reference</td>
<td>n/a</td>
<td>title of resource (reference: Dublin Core).</td>
</tr>
<tr>
<td>oslc:instanceShape</td>
<td>zero-or-one</td>
<td>True</td>
<td>Resource</td>
<td>Reference</td>
<td>n/a</td>
<td>instanceShape (<a href="http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#InstanceShape">http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#InstanceShape</a>) is an additional tag that can be applied to resources.</td>
</tr>
<tr>
<td>oslc: serviceProvider</td>
<td>many</td>
<td></td>
<td>Resource</td>
<td>Reference</td>
<td>n/a</td>
<td>serviceProvider (<a href="http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#ServiceProvider">http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#ServiceProvider</a>) is a resource that provides hints to a consumer about the nature of the target resource.</td>
</tr>
</tbody>
</table>

### Relationship properties: This grouping of properties is used to identify relationships between resources managed by OSLC Service Providers

<table>
<thead>
<tr>
<th>Prefixed 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>oslc_auto:parameterDefinition</td>
<td>zero-or-many</td>
<td></td>
<td>AnyResource</td>
<td>Either</td>
<td>any</td>
<td>parameterDefinition (<a href="http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#ParameterDefinition">http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#ParameterDefinition</a>) defines the parameter definitions of an Automation Plan.</td>
</tr>
<tr>
<td>oslc:usesExecutionEnvironment</td>
<td>zero-or-many</td>
<td></td>
<td>Resource</td>
<td>Reference</td>
<td>any</td>
<td>usesExecutionEnvironment (<a href="http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#UsesExecutionEnvironment">http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#UsesExecutionEnvironment</a>) specifies the environment(s) in which this Automation Plan can be executed.</td>
</tr>
<tr>
<td>oslc:futureAction</td>
<td>zero-or-many</td>
<td></td>
<td>Resource</td>
<td>Reference</td>
<td>any</td>
<td>futureAction (<a href="http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#FutureAction">http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#FutureAction</a>) defines actions that will become available on Automation Results that result from execution of this Plan.</td>
</tr>
</tbody>
</table>

**Resource: AutomationRequest**

- **Name:** AutomationRequest
- **Description:** A resource representing the intention to execute an Automation Plan. The Automation Request contains the information required to request that the provider execute an Automation Plan.
- **Type URI:** http://open-services.net/ns/auto#AutomationRequest

### AutomationRequest Properties

<table>
<thead>
<tr>
<th>Prefixed 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:contributor</td>
<td>zero-or-many</td>
<td></td>
<td>AnyResource</td>
<td>Either</td>
<td>any</td>
<td>contributor or contributors to resource (reference: Dublin Core). It is likely that the target resource will be an foaf:Person resource, but that is not necessarily the case.</td>
</tr>
<tr>
<td>dcterms:created</td>
<td>zero-or-one</td>
<td>True</td>
<td>DateTime</td>
<td>n/a</td>
<td>n/a</td>
<td>Timestamp of resource creation (reference: Dublin Core)</td>
</tr>
<tr>
<td>dcterms:creator</td>
<td>zero-or-many</td>
<td></td>
<td>AnyResource</td>
<td>Either</td>
<td>any</td>
<td>creator or creators of resource (reference: Dublin Core). It is likely that the target resource will be an foaf:Person resource, but that is not necessarily the case.</td>
</tr>
<tr>
<td>dcterms:propertyDefinition</td>
<td>zero-or-one</td>
<td>True</td>
<td>Resource</td>
<td>Reference</td>
<td>n/a</td>
<td>propertyDefinition (<a href="http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#PropertyDefinition">http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#PropertyDefinition</a>) serves the purpose of providing hints to a consumer about the nature of a property of the target resource.</td>
</tr>
<tr>
<td>rdf:type</td>
<td>zero-or-many</td>
<td></td>
<td>String</td>
<td>n/a</td>
<td>n/a</td>
<td>type of resource (reference: Dublin Core).</td>
</tr>
<tr>
<td>dcterms:subject</td>
<td>zero-or-many</td>
<td></td>
<td>String</td>
<td>n/a</td>
<td>n/a</td>
<td>subject of resource (reference: Dublin Core).</td>
</tr>
<tr>
<td>dcterms:title</td>
<td>exactly-one</td>
<td>True</td>
<td>Resource</td>
<td>Reference</td>
<td>n/a</td>
<td>title of resource (reference: Dublin Core).</td>
</tr>
<tr>
<td>oslc:instanceShape</td>
<td>zero-or-one</td>
<td>True</td>
<td>Resource</td>
<td>Reference</td>
<td>n/a</td>
<td>instanceShape (<a href="http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#InstanceShape">http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#InstanceShape</a>) is an additional tag that can be applied to resources.</td>
</tr>
<tr>
<td>oslc: serviceProvider</td>
<td>many</td>
<td></td>
<td>Resource</td>
<td>Reference</td>
<td>n/a</td>
<td>serviceProvider (<a href="http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#ServiceProvider">http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#ServiceProvider</a>) is a resource that provides hints to a consumer about the nature of the target resource.</td>
</tr>
</tbody>
</table>
Resource: AutomationResult

- **Name**: AutomationResult
- **Description**: A resource representing the intermediate and final execution state of an Automation Request, along with contributions to the result.
- **Type URI**: http://open-services.net/ns/auto#AutomationResult

**AutomationRunProperties**

<table>
<thead>
<tr>
<th>Prefixed 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>olscl:AutoRun</td>
<td>exactly-one</td>
<td>False</td>
<td>Resource</td>
<td>Reference</td>
<td>any</td>
<td>Automation Plan run by the Automation Request. It is likely that the target resource will be an Automation Request (whether by delegated UI or HTTP POST) and MAY include additional parameters added by the service provider during Automation Request creation.</td>
</tr>
<tr>
<td>olscl:AutoRun:state</td>
<td>exactly-one</td>
<td>False</td>
<td>Resource</td>
<td>Reference</td>
<td>any</td>
<td>Creator or contributors to resource (reference: Dublin Core). It is likely that the target resource will be an foaf:Person (<a href="http://open-services.net/ns/auto#Person">http://open-services.net/ns/auto#Person</a>), but that is not necessarily the case. The service provider or its agents is the creator of the resource, a foaf:Person (<a href="http://xmlns.com/foaf/spec/Perm-Agent">http://xmlns.com/foaf/spec/Perm-Agent</a>) could be used.</td>
</tr>
<tr>
<td>olscl:AutoRun:state:from</td>
<td>exactly-one</td>
<td>False</td>
<td>Resource</td>
<td>Reference</td>
<td>any</td>
<td>Timestamp of resource creation (reference: Dublin Core)</td>
</tr>
<tr>
<td>olscl:AutoRun:state:with</td>
<td>exactly-one</td>
<td>False</td>
<td>Resource</td>
<td>Reference</td>
<td>any</td>
<td>Title (reference: Dublin Core) of the resource represented as rich text in XHMTL content.</td>
</tr>
<tr>
<td>olscl:AutoRun:state:with:from</td>
<td>exactly-one</td>
<td>False</td>
<td>Resource</td>
<td>Reference</td>
<td>any</td>
<td>Resource Shape that provides hints as to resource property value-types and allowed values.</td>
</tr>
<tr>
<td>olscl:AutoRun:state:with:to</td>
<td>exactly-one</td>
<td>False</td>
<td>Resource</td>
<td>Reference</td>
<td>any</td>
<td>The scope of a resource is a link to the resource's OSLC Service Provider.</td>
</tr>
</tbody>
</table>

**OSLC Core: Common Properties**

<table>
<thead>
<tr>
<th>Prefixed 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>olscl:AutoRun:inputParameter</td>
<td>zero-or-many</td>
<td>True</td>
<td>AnyResource Either</td>
<td>Reference</td>
<td>n/a</td>
<td>Contributors to the resource (reference: Dublin Core). It is likely that the target resource will be an foaf:Person, but that is not necessarily the case. The service provider or its agents is the creator of the resource, a foaf:Person (<a href="http://xmlns.com/foaf/spec/Perm-Agent">http://xmlns.com/foaf/spec/Perm-Agent</a>) could be used.</td>
</tr>
<tr>
<td>olscl:AutoRun:inputParameter:from</td>
<td>zero-or-many</td>
<td>True</td>
<td>AnyResource Either</td>
<td>Reference</td>
<td>n/a</td>
<td>Timestamp of resource creation (reference: Dublin Core)</td>
</tr>
<tr>
<td>olscl:AutoRun:inputParameter:to</td>
<td>zero-or-many</td>
<td>True</td>
<td>AnyResource Either</td>
<td>Reference</td>
<td>n/a</td>
<td>Tag or keyword for a resource. Each occurrence of a dc:subject property denotes an additional tag for the resource.</td>
</tr>
<tr>
<td>olscl:AutoRun:inputParameter:with</td>
<td>zero-or-many</td>
<td>True</td>
<td>AnyResource Either</td>
<td>Reference</td>
<td>n/a</td>
<td>Title (reference: Dublin Core) of the resource represented as rich text in XHMTL content.</td>
</tr>
<tr>
<td>olscl:AutoRun:inputParameter:with:from</td>
<td>zero-or-many</td>
<td>True</td>
<td>AnyResource Either</td>
<td>Reference</td>
<td>n/a</td>
<td>Resource Shape that provides hints as to resource property value-types and allowed values.</td>
</tr>
<tr>
<td>olscl:AutoRun:inputParameter:with:to</td>
<td>zero-or-many</td>
<td>True</td>
<td>AnyResource Either</td>
<td>Reference</td>
<td>n/a</td>
<td>The scope of a resource is a link to the resource's OSLC Service Provider.</td>
</tr>
</tbody>
</table>

**OSLC Automation:**

Start of additional properties

- **Predefined Name**: olscl:AutoRun
- **Occurs**: exactly-one
- **Value-type**: AnyResource
- **Description**: A unique identifier for a resource. Assigned by the service provider when a resource is created. Not intended for end-user display.

- **Predefined Name**: olscl:AutoRun:state
- **Occurs**: exactly-one
- **Value-type**: AnyResource
- **Description**: The resource type URIs.

- **Predefined Name**: olscl:AutoRun:desiredState
- **Occurs**: exactly-one
- **Value-type**: AnyResource
- **Description**: The title (reference: Dublin Core) of the resource represented as rich text in XHMTL content.

- **Predefined Name**: olscl:AutoRun:verdict
- **Occurs**: exactly-one
- **Value-type**: AnyResource
- **Description**: Resource Shape that provides hints as to resource property value-types and allowed values.

- **Predefined Name**: olscl:AutoRun:state:from
- **Occurs**: exactly-one
- **Value-type**: AnyResource
- **Description**: A copy of the parameters provided during creation of the Automation Request which produced this Automation Result (see olscl:AutoRun:producedBy:AutomationRequest). The olscl:AutoRun:inputParameter resources on an Automation Result should be considered a point-in-time copy of the parameter at the time the Automation Request was created.

- **Predefined Name**: olscl:AutoRun:state:to
- **Occurs**: exactly-one
- **Value-type**: AnyResource
- **Description**: Automation Result output parameters are parameters associated with the automation execution which produced this Result. This includes the final value of all parameters used to initiate the execution and any additional parameters which may have been created during automation.
### ParameterInstance Properties

<table>
<thead>
<tr>
<th>Prefixes Name</th>
<th>Occurs</th>
<th>Read-only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslc:Name</td>
<td>exactly-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>The name of the parameter instance.</td>
</tr>
<tr>
<td>rdf:value</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>The value of the parameter. The value may be an RDF literal or a resource. If the value is an RDF literal, then it SHOULD be an RDF typed literal.</td>
</tr>
<tr>
<td>dcterms:title</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>Descriptive text (reference: Dublin Core) about resource represented as rich text in XHTML content. SHOULD include only content that is valid and suitable inside an XHTML &lt;!DOCTYPE &gt; element.</td>
</tr>
<tr>
<td>rdf:type</td>
<td>zero-or-many</td>
<td>n/a</td>
<td>Reference n/a</td>
<td>Resource n/a</td>
<td>The resource type URIs.</td>
</tr>
<tr>
<td>oslc:InstanceShape</td>
<td>zero-or-one</td>
<td>True</td>
<td>Resource</td>
<td>ResourceShape</td>
<td>Resource Shape that provides hints as to resource property value-types and allowed values.</td>
</tr>
<tr>
<td>oslc:ServiceProvider</td>
<td>zero-or-many</td>
<td>True</td>
<td>Reference</td>
<td>n/a</td>
<td>The scope of a resource is a link to the resource’s OSLC Service Provider.</td>
</tr>
</tbody>
</table>

### Dialog

<table>
<thead>
<tr>
<th>Prefixes Name</th>
<th>Occurs</th>
<th>Read-only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcterms:title</td>
<td>exactly-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>Title string that could be used for display.</td>
</tr>
<tr>
<td>oslc:label</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>Very short label for use in menu items.</td>
</tr>
<tr>
<td>rdf:resource</td>
<td>exactly-one</td>
<td>n/a</td>
<td>Reference Any</td>
<td>N/A</td>
<td>The URI of the dialog.</td>
</tr>
<tr>
<td>dcterms:label</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>Reference Any</td>
<td>N/A</td>
<td>Values MUST be expressed as described in OSLC Core 2.0.</td>
</tr>
<tr>
<td>rdf:label</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>Reference Any</td>
<td>N/A</td>
<td>Values MUST be expressed as described in OSLC Core 2.0.</td>
</tr>
<tr>
<td>oslc:resourceType</td>
<td>zero-or-many</td>
<td>n/a</td>
<td>Resource Any</td>
<td>N/A</td>
<td>The expected resource type URI(s) for the resources that will be returned when using this dialog. These would be the URIs found in the result resource’s rdf:type property. In a deferred execution creation dialog, clients will expect at least one of these to be oslc:auto:CreationRequest.</td>
</tr>
<tr>
<td>oslc:usage</td>
<td>one-or-many</td>
<td>n/a</td>
<td>Resource Any</td>
<td>N/A</td>
<td>An identifier URI for the domain specified usage of this dialog. For example, for a deferred execution creation dialog this will be oslc:auto:DeferredExecution.</td>
</tr>
</tbody>
</table>

### OSLC Core: Common Properties

<table>
<thead>
<tr>
<th>Prefixes Name</th>
<th>Occurs</th>
<th>Read-only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslc:Name</td>
<td>exactly-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>The name of the parameter instance.</td>
</tr>
<tr>
<td>rdf:value</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>The value of the parameter. The value may be an RDF literal or a resource. If the value is an RDF literal, then it SHOULD be an RDF typed literal.</td>
</tr>
<tr>
<td>dcterms:title</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>Descriptive text (reference: Dublin Core) about resource represented as rich text in XHTML content. SHOULD include only content that is valid and suitable inside an XHTML &lt;!DOCTYPE &gt; element.</td>
</tr>
<tr>
<td>rdf:type</td>
<td>zero-or-many</td>
<td>n/a</td>
<td>Reference n/a</td>
<td>Resource n/a</td>
<td>The resource type URIs.</td>
</tr>
<tr>
<td>oslc:InstanceShape</td>
<td>zero-or-one</td>
<td>True</td>
<td>Resource</td>
<td>ResourceShape</td>
<td>Resource Shape that provides hints as to resource property value-types and allowed values.</td>
</tr>
<tr>
<td>oslc:ServiceProvider</td>
<td>zero-or-many</td>
<td>True</td>
<td>Reference</td>
<td>n/a</td>
<td>The scope of a resource is a link to the resource’s OSLC Service Provider.</td>
</tr>
</tbody>
</table>

This is new for 2.1: START

<table>
<thead>
<tr>
<th>Prefixes Name</th>
<th>Occurs</th>
<th>Read-only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslc:Name</td>
<td>exactly-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>The name of the parameter instance.</td>
</tr>
<tr>
<td>rdf:value</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>The value of the parameter. The value may be an RDF literal or a resource. If the value is an RDF literal, then it SHOULD be an RDF typed literal.</td>
</tr>
<tr>
<td>dcterms:title</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>Descriptive text (reference: Dublin Core) about resource represented as rich text in XHTML content. SHOULD include only content that is valid and suitable inside an XHTML &lt;!DOCTYPE &gt; element.</td>
</tr>
<tr>
<td>rdf:type</td>
<td>zero-or-many</td>
<td>n/a</td>
<td>Reference n/a</td>
<td>Resource n/a</td>
<td>The resource type URIs.</td>
</tr>
<tr>
<td>oslc:InstanceShape</td>
<td>zero-or-one</td>
<td>True</td>
<td>Resource</td>
<td>ResourceShape</td>
<td>Resource Shape that provides hints as to resource property value-types and allowed values.</td>
</tr>
<tr>
<td>oslc:ServiceProvider</td>
<td>zero-or-many</td>
<td>True</td>
<td>Reference</td>
<td>n/a</td>
<td>The scope of a resource is a link to the resource’s OSLC Service Provider.</td>
</tr>
</tbody>
</table>

### OSLC Core: Common Properties

<table>
<thead>
<tr>
<th>Prefixes Name</th>
<th>Occurs</th>
<th>Read-only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslc:Name</td>
<td>exactly-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>The name of the parameter instance.</td>
</tr>
<tr>
<td>rdf:value</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>The value of the parameter. The value may be an RDF literal or a resource. If the value is an RDF literal, then it SHOULD be an RDF typed literal.</td>
</tr>
<tr>
<td>dcterms:title</td>
<td>zero-or-one</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>Descriptive text (reference: Dublin Core) about resource represented as rich text in XHTML content. SHOULD include only content that is valid and suitable inside an XHTML &lt;!DOCTYPE &gt; element.</td>
</tr>
<tr>
<td>rdf:type</td>
<td>zero-or-many</td>
<td>n/a</td>
<td>Reference n/a</td>
<td>Resource n/a</td>
<td>The resource type URIs.</td>
</tr>
<tr>
<td>oslc:InstanceShape</td>
<td>zero-or-one</td>
<td>True</td>
<td>Resource</td>
<td>ResourceShape</td>
<td>Resource Shape that provides hints as to resource property value-types and allowed values.</td>
</tr>
<tr>
<td>oslc:ServiceProvider</td>
<td>zero-or-many</td>
<td>True</td>
<td>Reference</td>
<td>n/a</td>
<td>The scope of a resource is a link to the resource’s OSLC Service Provider.</td>
</tr>
</tbody>
</table>

This is new for 2.1: END
Automation Service Provider Capabilities

Asynchronous and Synchronous Automation Execution

An OSLC Automation service provider is generally assumed to implement automation requests asynchronously. In this model, a client creates an automation request and then later queries a collection of automation results for the particular result(s) related to its request. For generality, it is also assumed that results may be contributed asynchronously by a set of distributed processes, where each contributor adds its contribution(s) to the result via HTTP PUT. When a provider creates an automation request, it may also include an automation result.

Delegated UIs

An instance of an OSLC Automation service provider might provide services for one or more particular automation sub-domains (e.g. test or build automation). Automation service providers MAY declare sub-domain information in the Service Provider document by specifying a sub-domain value in the oslc:usage attribute on the oslc:Service resource in the Service Provider document. Valid sub-domain values are:

- http://open-services.net/ns/auto#Build: Indicates that the related service provider or services provide build automation capabilities - the process of converting source code artifacts into software artifacts such as executables, libraries and documentation.
- http://open-services.net/ns/auto#Test: Indicates that the related service provider or services provide test automation capabilities - the process of executing tests on a system under test and comparing the results of the tests to pass/fail conditions.
- http://open-services.net/ns/auto#Deploy: Indicates that the related service provider or services provide deployment capabilities - the process of executing processes and procedures to ready systems and software for use.

An automation service provider which is a general-purpose automation provider, or a provider not wanting to provide a sub-domain should provide an oslc:usage value of "http://open-services.net/ns/auto". If no oslc:usage attribute indicating a sub-domain is present, the default is assumed to be "http://open-services.net/ns/auto".

Sub-domain Example

Example of a service provider document fragment with a 2 Services which are identified as related to the Test and Deploy sub-domains:

```xml
<oslc:serviceProvider>
  <oslc:service>
    <oslc:usage rdf:resource="http://open-services.net/ns/auto#Test"/>
    <oslc:queryCapability>
      ...
    </oslc:queryCapability>
    <oslc:creationFactory>
      ...
    </oslc:creationFactory>
  </oslc:service>
  <oslc:service>
    <oslc:usage rdf:resource="http://open-services.net/ns/auto#Deploy"/>
    <oslc:queryCapability>
      ...
    </oslc:queryCapability>
    <oslc:creationFactory>
      ...
    </oslc:creationFactory>
  </oslc:service>
</oslc:serviceProvider>
```

Resource Shapes

OSLC Automation service providers MAY support Resource Shapes (http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA#oslc_ResourceShape_Resource) as defined in OSLC Core Specification Appendix A (http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA)

Service Provider Resource

OSLC Automation service providers MUST provide a Service Provider Resource (http://open-services.net/bin/view/Main/OslcCoreSpecification#Service_Provider_Resources) that can be retrieved at a implementation dependent URI.

OSLC Automation service providers MAY provide a Service Provider Catalog Resource (http://open-services.net/bin/view/Main/OslcCoreSpecifications#Resource_Service_Provider_Catalog) that can be retrieved at a implementation dependent URI.

It is RECOMMENDED that OSLC Automation service providers provide a oslc:serviceProvider property for their defined resources that will be the URI to a Service Provider Resource (http://open-services.net/bin/view/Main/OslcCoreSpecifications#Service_Provider_Resources).

Creation Factories

If an OSLC Automation service provider supports the creation of resources, there MUST be at least one Creation Factories (http://open-services.net/bin/view/Main/OslcCoreSpecifications#Creation_Factories) entry in the Services definition. See HTTP Method support table for further clarification on support for HTTP methods and media types for each OSLC Automation resource.

Query Capabilities

OSLC Automation service providers SHOULD have at least one Query Capabilities entry in the its Services definition that allows a client to query AutomationResults.

Note: OSLC Automation does not require providers to keep resources accessible forever. Clients should not expect automation results to be available for any particular length of time once the request has finished. Some providers might respond to an AutomationRequest creation request with an AutomationResult that is also an AutomationResult, and might make the result inaccessible immediately thereafter.

Note: If an OSLC Automation provider does expose a Query Capability that applies to AutomationResults, and if its AutomationRequest creation responses are not also AutomationResults, then its Query Capability is the only Automation-defined way for clients to find the corresponding AutomationResults.

The Query Capability MUST support these OSLC query parameters and MAY support others:

- oslc:where
- oslc:select

If shape information is NOT present with the Query Capability, service providers SHOULD use the default properties defined in OSLC Core RDF/XML Examples (http://open-services.net/bin/view/Main/OslcCoreSpecRDFXMLExamples#Specifying_the_shape_of_a_query) to contain the result.

Selective Property Values

OSLC Automation providers SHOULD support the oslc:properties syntax for selective property value retrieval when a resource is accessible via its resource URI.

Delegated UIs

OSLC Automation service providers support the selection and creation of Automation resources as defined by Delegated UIs (http://open-services.net/bin/view/Main/OslcCoreSpecifications#Delegated_User_Interface_Dialogs) in OSLC Core.

The service providers supports requirements for delegated UIs is as follows:

**Automation Resource Selection Creation**

AutomationPlan SHOULD MAY
AutomationRequest MAY SHOULD
AutomationResult SHOULD MAY
Immediate-Execution Creation Dialog

An "immediate-execution" creation dialog is one that creates an Automation Request and makes it available for execution as soon as it is created. This is the only form of creation dialog that was defined in OSLC Automation 2.0. OSLC Automation 2.1 defines the term "immediate-execution creation dialog" and the oslc:usage URI http://open-services.net/ns/auto#ImmediateExecution (oslc_auto:ImmediateExecution) to distinguish them from deferred-execution creation dialogs.

OSLC Automation 2.1 consumers MUST interpret an oslc_auto:AutomationRequest creation dialog that has neither oslc:usage oslc_auto:ImmediateExecution set nor oslc:usage oslc_auto:DeferredExecution set as being an "immediate-execution creation dialog". This is to maintain compatibility with OSLC Automation 2.0 providers.

If an OSLC Automation provider offers both immediate-execution and deferred-execution creation dialogs, it MUST provide oslc_auto:ImmediateExecution or oslc_auto:DeferredExecution as a oslc:usage value (respectively) on the oslc:Dialog creation dialog resources. Such a provider SHOULD also set oslc:default as a oslc:usage value on the immediate-execution dialogs, to guide OSLC Automation 2.0 consumers to use those dialogs and not the deferred-execution ones.

Deferred-Execution Creation Dialog

A Deferred-Execution Creation Dialog is a resource creation deferred user interface dialog (bin/view/Main/OsdcCoreSpecification#Resource_creation_in_a_dd) that creates an Automation Request but does not make it eligible for execution. A deferred-execution creation dialog MUST comply with all Core requirements on resource creation deferred user interface dialogs (bin/view/Main/OsdcCoreSpecification#Resource_creation_in_a_dd). One important consequence of this is that all facilities available on resource creation deferred user interface dialogs, for example the https://bin/view/Main/OsdcCoreSpecification#PreFiller, Creation_Dialogs, apply equally to deferred-execution creation dialogs.

This specification defines the oslc:usage URI http://open-services.net/ns/auto#DeferredExecution (oslc_auto:DeferredExecution) to allow clients to discover Automation Request creation dialogs created by an Automation provider supplies, amongst any other dialogs in their oslc:Service resources, as shown in this example /wiki/automation/OSLC-Automation-Version-2.1-Samples/#deferred-and-immediate-dialog-provider. The corresponding resource shape is shown in an earlier section.

Non-normative note: The Automation 2.0 specification only provided a standard way to create Automation Requests that are eligible for execution once they are created; Automation 2.1 adds deferred-execution creation dialogs to allow creation without execution eligibility in a standard way. This meets Automation 2.1’s template scenarios (/wiki/automation/Automation-Scenarios-v2.1), while retaining compatibility with Automation 2.0 clients by keeping the behavior of oslc:creationDialog resources unchanged from 2.0.

Non-normative note: we suggest that providers allow these resources to exist for at least 15 minutes, but the actual value used is implementation-dependent.

A deferred-execution creation dialog is especially likely to be short-lived (cleaned up by the server shortly after creation); while this can be true of resources in general, for historical reasons (the 2.0 creation factory behavior described above) it is particularly important in this case as a common usage pattern. As a consequence, the consumer SHOULD get its representation immediately after creating it.

Deferred-Execution Creation Dialog

A Deferred-Execution Creation Dialog is a resource creation deferred user interface dialog (bin/view/Main/OsdcCoreSpecification#Resource_creation_in_a_dd) that creates an Automation Request but does not make it eligible for execution. A deferred-execution creation dialog MUST comply with all Core requirements on resource creation deferred user interface dialogs (bin/view/Main/OsdcCoreSpecification#Resource_creation_in_a_dd). One important consequence of this is that all facilities available on resource creation deferred user interface dialogs, for example the https://bin/view/Main/OsdcCoreSpecification#PreFiller, Creation_Dialogs, apply equally to deferred-execution creation dialogs.

When a deferred-execution creation dialog creates an Automation Request, it is not queued for execution unless the client takes some explicit further action; it is the responsibility of the consumer to decide when (if ever) it is ready to be executed. OSLC defines options to initiate execution that include the following:

- Provide it as input to a standard (immediately-execution-eligible) Automation creation factory (bin/view/Main/OsdcCoreSpecification#Creation_Factories).
- Provide it as input to a standard (immediately-execution-eligible) Automation creation dialog that supports pre-fill.

Note: assuming that the request is successful, it is important to recognize that the cases above all result in the creation of a new Automation Request, with a different URI than anything provided as an input. The provider may provide other ways, in addition to or in place of these, for the consumer to use when it is ready to have the Automation Request executed. OSLC currently has no scenarios requiring the definition of a way to change the state to make the same (input) request eligible for execution.

OSLC defines options for locating those immediate-execution resources, for example creation factories and delegated creation dialogs, that include the following:

- Consumers can examine an OSLC Service Provider document's oslc:Service resources. In many scenarios, Automation clients will only need to implement the Creation Factory interaction pattern to initiate execution, although other possibilities exist.
- Consumers can use oslc:binding properties on the oslc_auto:DeferredExecution dialog resource to simplify the process of locating appropriate immediate-execution resources. Those consumers choose at least one Actions specification profile, and implement the interaction patterns described in that profile. They are only able to consume deferred-execution dialogs whose bindings use an interaction pattern that the consumer implements.

The Automation provider MUST describe how to immediately execute an Automation Request created by a deferred-execution dialog using one or more oslc:binding properties on the oslc_auto:DeferredExecution dialog resource. If the deferred-execution dialog is discoverable from a Service in a Service Provider (bin/view/Main/OsdcCoreSpecification#Service_Provider_Resources), then the provider MUST supply at least one immediate-execution binding whose target uses the Automation creation factory interaction pattern. If multiple oslc:binding properties are present, they MUST be equivalent alternatives to each other, as defined by Core Actions.

When the second class of consumer from the list above is ready to execute an Automation Request acting as a template, it uses one of the oslc:binding properties on the deferred-execution dialog to immediately execute the action (often, by creating a new Automation Request with a different URI). The consumer does this by following the selected binding's instructions (/wiki/core/Actions-/2.0/); its interaction pattern might be defined by this specification, or might be defined by another specification. A consumer chooses which oslc:binding property to use based on which interaction patterns it understands. If there are no oslc:binding values whose interaction patterns are understood by the consumer then the Automation Request acting as a template is discarded by this consumer and the consumer SHOULD indicate this to the user instead of allowing them to use the deferred-execution dialog. A full example (/wiki/automation/OSLC-Automation-Version-2.1-Samples/#deferred-execution-binding) is available in the companion Samples document.

Execution environments

An AutomationPlan can use the oslc_auto:usesExecutionEnvironment predicate to link to a resource representing the environment(s) that Automation Plan can be executed in. The execution environment resource could represent a grouping of environmental details such as operating system, database, browser, compiler, etc. The type of that resource, and the predicates to use on it, are not defined by this specification.

If more than one execution environment is specified on the Automation Plan, the consumer is expected to specify the desired execution environment as part of the Automation Request which it is constructing for the Automation Plan’s execution. The execution environment is provided as an InputParameter to the Automation Request.

The consumer is expected to find a parameter definition from the Automation Plan with oslc:propertyDefinition property set to http://open-services.net/ns/auto#ExecutionEnvironment, and to create an InputParameter on the Automation Request for that parameter definition, specifying the execution environment to use (choosing out of those specified on the Automation Plan). If that parameter definition’s oslc:occurs property is exactly-one or one-or-more, then the consumer MUST specify an execution environment. Otherwise, the consumer MAY specify an execution environment.

This is new for 2.1; END

State and Verdict properties

OSLC Automation service providers can identify the state and verdict using references to property values in the OSLC Automation vocabulary or to property values that are not in the Automation vocabulary (i.e. in the service provider’s own vocabulary). It is expected that the state and verdict values will be URI references to property values, but inline resources defining the state and verdict property values are also valid. Automation service providers MUST use at least one verdict (Automation Results) and state (Automation Requests and Results) defined in the OSLC automation vocabulary in addition to any states or verdicts not in the Automation vocabulary.

The additional property values for oslc_auto:state are:

- http://open-services.net/ns/auto#new - used to indicate an automation request or result has just been created in the service provider and has not yet been acted upon.
- http://open-services.net/ns/auto#queued - primarily used to indicate an automation request or result is queued for additional actions by the service provider.
- http://open-services.net/ns/auto#inProgress - used to indicate an automation request or result is active in the service provider.
- http://open-services.net/ns/auto#canceled - used to indicate that an automation request or result has been canceled.
- http://open-services.net/ns/auto#complete - used to indicate that an automation request or result is complete.

The additional property values for oslc_auto:verdict are:

- http://open-services.net/ns/auto#unavailable - used to indicate an automation result is in a state where a final verdict such as oslc_auto:passed or oslc_auto:failed is not yet available. Usually used when the result is in a state other than oslc_auto:complete.
OSLC Actions and Automation

This specification defines extensions to the OSLC Actions 2.0 specification (wiki/core/Actions-2.0). Actions provide “a means of advertising actions (or operations) that can be performed on (or in the context of) a specific resource”. This relates to Automation in two ways: firstly, Automation Requests can be used as an interaction pattern by which actions can be executed, and secondly, Actions can provide a way to aid management and the lifecycle of automation resources.

The Actions specification reuses Automation resources to define an Automation Request interaction pattern, which can be used to execute actions. Actions also defines a specification profile that implementations can use, which provides interoperability based on providers and consumers both using a common interaction pattern. This specification extends the Actions specification by defining interaction patterns which are useful in the management of automation resources.

See also: Deciding how to use Actions and Automation together

Discovering actions and choosing bindings

Discovering executable actions and choosing bindings

For information on how to discover currently-available actions on resources and how to choose which binding to use for execution, see the OSLC Actions 2.0 specification (wiki/core/Actions-2.0).

Discovering actions that will be executable after an Automation Request completes

One Automation use of Core’s actions is to advertise actions that become available after an Automation Request completes: for example, tearing down a deployed system, promoting or deleting a build. If the execution of the Automation Request resulted in a new resource being created (e.g. a resource representing the deployed system, or a resource representing the build) then it is expected that newly created resource would be linked to as an oslc:Action resource on the Automation Result, and any action in the context of that new resource would be linked to as an oslc:Action on that resource. However, consumers may not know which characteristics to check for action, so any actions that would make sense to follow up the execution of an Automation Request - whether immediately or at a later time - SHOULD be advertised on the Automation Result in addition to (or instead of) a contribution.

Future actions

It is sometimes useful to know what actions will be available before an Automation Request is created (for example, for scheduling automated processes that will execute in their entirety without user intervention). Users might prefer such Automation Plans over otherwise equivalent ones that lack the ability to automate cleanup after themselves, so in fact it can be useful to know about future actions when selecting a Plan, before any Automation Request has even been created. Note: Core Actions (wiki/core/Exposing-arbitrary-actions-on-RDF-resources/#Future-actions) mentions other uses of future actions.

To enable cases like these, providers MAY link to future actions using the oslc:futureAction predicate. When an Automation provider provides future action link(s) on an Automation Plan, they SHOULD link to resources of type oslc:Action which describe an action that may be executed after an execution of the Automation Plan has completed. As such, these oslc:Action resources SHOULD NOT contain any bindings that can be executed immediately.

Non-normative note: Bindings using the “deferred execution dialog interaction pattern” may be present, but this specification does not define how to use them for future actions. It would not make sense to specify a deferred execution dialog execution binding for a future action, because its mandatory immediate-execution binding cannot become available until after an Automation Request has been created.

These future action resources describe what kind of actions are available on the Automation Result, so consumers can present these to users in preparation for when the execution has completed, and so the oslc:Action resources SHOULD include all the properties needed to render a display of the action. These oslc:Action resources SHOULD NOT be anonymous (RDF blank) nodes, so they can be linked to by the executable actions on the results using the predicate below.

The execution of these future actions requires an immediately executable action on an Automation Result. When an Automation Plan containing future actions is executed, each action applicable to the generated result SHOULD have an equivalent immediately executable action, linked to using the oslc:Action predicate, from the Automation Result. Each of these actions SHOULD use the oslc:executes predicate to link to the future action on the Plan that it relates to. This allows consumers to map a user selection of a future action on the Plan to an executable action on the result. Each future action SHOULD have at least one executable action linking to it from each Result. (Note: If a Plan’s future action PFA specifies a binding using the deferred execution dialog interaction pattern, then the corresponding Result’s action bindings linking back to PFA might be intended as immediate-execution bindings for the deferred execution dialog (see below), but this specification does not require that usage).

See the Temporary-deployment-scenarios (wiki/automation/Temporary-deployment-scenarios/) for a worked example of future actions.

Deciding how to use Actions and Automation together

This section is non-normative.

When implementing a provider of Automation Plans, you can decide whether to expose those plans through Actions or not. This section addresses that decision.

There are two main issues that come into play: discovery and execution. In the Automation 2.0 specification, which predated the OSLC Actions specification, Automation Plans were discovered through query capabilities or selection dialogs on a service provider. This was the only way to discover them. Actions provide an additional option for discovery, in the context of any given resource. That is, if a given Automation Plan "acts on" another resource, it makes sense for that resource to point to that Automation Plan, including information on what executing that plan will achieve. (Plans discovered via Actions can still be made discoverable through the normal means as well, for consumers who don’t want to browse other resources, but instead just want to directly list or select an Automation Plan).

Automation Plans have a well-defined means of requesting execution. Automation Plans are one option for how providers can allow their actions to be executed. However, unlike plain Automation Plans discovered from a query capability or selection dialog, actions allow providers to specify other means of execution in addition to or instead of Automation Plans (while still supporting predictable interoperability being implemented against "specification profiles"). See the information on "interaction patterns" and "specification profiles" in the OSLC Actions 2.0 specification for more information.

<table>
<thead>
<tr>
<th>Automation Plans only</th>
<th>Query capabilities/Selection dialogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions</td>
<td>On other resources (which will be the context of the execution)</td>
</tr>
<tr>
<td>(Actions’ Automation Plans can also be made available through query capabilities or selection dialogs as with other plans)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Discovery</th>
<th>Creation of Automation Request</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Execution</th>
<th>Creation of Automation Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Actions can also provide other non-Automation Plan bindings that the consumer can choose as an alternative)</td>
<td></td>
</tr>
</tbody>
</table>

OSLC Actions Extensions

Teardown action type
This specification defines the RDF class `oslc_auto:TeardownAction`, as an rdfs:subClassOf `oslc:Action`, with the meaning that any action of this type MUST have the semantics of tearing down some deployed resource. It is likely that such a deployed resource was deployed using an OSLC Automation deployment plan, but this MAY not be the case. That is, a tear-down action typically has the opposite semantics from a `oslc_auto:Deploy` sub-domain Automation Plan or Request, even if the service provider offers no equivalents in its Automation Plan collection.

**Automation Request interaction pattern**

This interaction pattern is defined by the OSLC Actions 2.0 specification [http://open-services.net/wiki/core/Actions-2.0/](http://open-services.net/wiki/core/Actions-2.0/) (for reuse by other domain specifications).

**Automation Creation Factory interaction pattern**

This section defines how to use an OSLC Core Creation Factory [http://open-services.net/bin/view/Main/OslcCoreSpecification#Creation_Factories](http://open-services.net/bin/view/Main/OslcCoreSpecification#Creation_Factories) that creates OSLC Automation Requests eligible for immediate execution as an Actions 2.0 [http://open-services.net/wiki/core/Actions-2.0/](http://open-services.net/wiki/core/Actions-2.0/) interaction pattern.

**Pattern recognition rule**

For any action binding that uses this interaction pattern:

- at least one `rdfs:property` MUST have the value `oslc:CreationFactory`.
- at least one `oslc:resourceType` property MUST have the value `oslc_auto:AutomationRequest`.
- at least one `oslc:usage` property MUST have the value `oslc_auto:ImmediateExecution`.
- the `oslc:finalStatusLocation` property MUST have the value `oslc_auto:AutomationResult`.

A binding is deemed to use this pattern if it meets these restrictions.

**Deferred execution dialog interaction pattern**

This section defines the `Deferred-Execution Creation Dialog` interaction pattern as an Actions 2.0 [http://open-services.net/wiki/core/Actions-2.0/](http://open-services.net/wiki/core/Actions-2.0/) interaction pattern designed to address scheduling scenarios. This interaction pattern consists of the following stages:

1. **Configuration**: The consumer displays a deferred-execution creation dialog to a user for them to configure an action. An arbitrary time delay occurs. This accommodates use cases like calendar-schedule execution and manual approval cycles.
2. **Execution**: One or more executions of the configured action. Each execution uses a new resource with a separate lifecycle from the previously configured action, and might either require a user (to supply final configuration values) or might be fully automated.

In this interaction pattern, the consumer is in charge of when the action is executed. (If the provider needs to be in charge of when the action becomes executable, the standard “delegated UI dialog for immediate execution” interaction pattern can be used, with provider exercising whatever degree of control it needs to; for example, creating it immediately and internally holding it, or deferring its creation internally.)

**Pattern recognition rule**

For any action binding that uses this interaction pattern:

- at least one `rdfs:property` MUST have the value `oslc:Dialog`.
- at least one `oslc:usage` property MUST have the value `oslc_auto:DeferredExecution`.
- the `oslc:finalStatusLocation` property MUST have the value `oslc:Dialog`.

A binding is deemed to use this pattern if it meets these restrictions.
To execute an action binding using this interaction pattern, a consumer does the following:

1. **Configuration stage**
   - The consumer follows the requirements in the OSLC Core Delegated UI specification [http://open-services.net/wiki/core/Actions-2.0/#pattern-immed-dialog] to display the deferred-execution creation dialog (recall that deferred-execution creation dialogs are also standard creation dialogs). The dialog will either return a URI or an error code, which gives the client the status of this phase of its goal.
   - If the dialog returns a URI, then the consumer performs an HTTP GET request on that URI immediately and stores the result representation to be used at the later time to execute the action.

2. **Execution stage**
   - If and when the consumer comes to execute the action at a later time, then the consumer selects an interaction pattern and follows its instructions, but with the changes described under immediate-execution bindings below.

Immediate-execution bindings

The delegated UI dialog for later execution interaction pattern involves two bindings: one at configuration time that creates the configuration for use at the later time, and a second binding that once executed (with the configuration returned from the first binding) triggers the action immediately. Hence, these second bindings are called "immediate-execution bindings".

Immediate-execution bindings MAY use any of the following interaction patterns for the execution of this interaction pattern. In each case, the input representation MUST be replaced by the representation saved during the configuration stage, regardless of whether it is used as a request body, dialog pre-fill, or other purpose by the patterns listed below.

- HTTP request with Resource Shape to describe the request body ([/wiki/core/Actions-2.0/#pattern-resource-shape](http://open-services.net/wiki/core/Actions-2.0/#pattern-resource-shape))
- HTTP request with fixed body ([/wiki/core/Actions-2.0/#pattern-body-runtime](http://open-services.net/wiki/core/Actions-2.0/#pattern-body-runtime))
- Automation Request ([/wiki/core/Actions-2.0/#pattern-autoreq](http://open-services.net/wiki/core/Actions-2.0/#pattern-autoreq))
- Delegated UI dialog for immediate execution ([/wiki/core/Actions-2.0/#pattern-immed-dialog](http://open-services.net/wiki/core/Actions-2.0/#pattern-immed-dialog))
- Automation Creation Factory

Consumers MUST NOT use these interaction patterns on immediate-execution bindings, even if the binding meets the pattern’s recognition rule:

- HTTP request with empty body ([/wiki/core/Actions-2.0/#pattern-empty-body](http://open-services.net/wiki/core/Actions-2.0/#pattern-empty-body)) cannot be used because there is no conceptual input representation slot.
- Deferred execution dialog interaction pattern cannot be used because execution is deferred.

Other specifications that define new interaction patterns MAY state whether or not those interaction patterns can be used as immediate-execution bindings, and if they are allowed, then how to use the template to execute them.

This is new for 2.1-END

### Automation Service Provider HTTP method support

For V2 of the OSLC Automation specification, support for all HTTP methods in the compliance table is **not** required for all Automation resources. The following table summarizes the requirements for each resource type, HTTP method and for each media type.

<table>
<thead>
<tr>
<th>Resource</th>
<th>RDF/XML</th>
<th>XML</th>
<th>JSON</th>
<th>OSLC</th>
<th>HTML</th>
<th>Unspecified</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Automation Plan</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>MUST</td>
<td>MAY</td>
<td>SHOULD</td>
<td>SHOULD</td>
<td>SHOULD</td>
<td>N/A</td>
</tr>
<tr>
<td>PUT</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POST</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DELETE</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>MAY</td>
</tr>
<tr>
<td><strong>Automation Request</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>MUST</td>
<td>MAY</td>
<td>MAY</td>
<td>SHOULD</td>
<td>SHOULD</td>
<td>N/A</td>
</tr>
<tr>
<td>PUT</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POST</td>
<td>MUST</td>
<td>MAY</td>
<td>SHOULD</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DELETE</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>MAY</td>
</tr>
<tr>
<td><strong>Automation Result</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>MUST</td>
<td>MAY</td>
<td>MAY</td>
<td>SHOULD</td>
<td>SHOULD</td>
<td>N/A</td>
</tr>
<tr>
<td>PUT</td>
<td>SHOULD</td>
<td>MAY</td>
<td>SHOULD</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POST</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DELETE</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>MAY</td>
</tr>
<tr>
<td><strong>Parameter Definition</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
</tr>
<tr>
<td>PUT</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POST</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DELETE</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>MAY</td>
</tr>
<tr>
<td><strong>Parameter Instance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
</tr>
<tr>
<td>PUT</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POST</td>
<td>MAY</td>
<td>MAY</td>
<td>MAY</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DELETE</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>MAY</td>
</tr>
</tbody>
</table>

OSLC Automation service providers **SHOULD** support deletion of any resources for which it allows creation.

### Automation Specification Guidance

This section is non-normative, i.e. it does not affect compliance.

#### Canceling the execution of an automation request

The **Automation Request** and **Automation Result** resources have an `oslc_auto:desiredState` attribute. A consumer can indicate a desire to cancel the execution of an automation by doing a PUT to the artifact with this attribute set to a value of `http://open-services.net/ns/auto#canceled`. If the service provider supports cancelation of automation executions, the receipt of a PUT with this attribute set should trigger the necessary provider processing. If the cancelation is successful, the service provider should set the appropriate artifact `oslc_auto:state` to `http://open-services.net/ns/auto#canceled`.

- When only an Automation Request is active (Automation Result not created yet), the consumer should request cancelation by setting `oslc_auto:desiredState` to `http://open-services.net/ns/auto#canceled` on the Automation Request.
- When Automation Requests and Automation Results are active (in an `oslc_auto:state` other than `oslc_auto:canceled` or `oslc_auto:complete`), the consumer should request cancelation by setting `oslc_auto:desiredState` to `http://open-services.net/ns/auto#canceled` on the Automation Request.
- When only an Automation Result is active (Automation Request completed, canceled or no longer exists), the consumer should request cancelation by setting `oslc_auto:desiredState` on the Automation Result.
- Consumers are responsible for checking the status code of the response to the request for cancelation and for checking the `oslc_auto:state` of the resource.
If a service provider does not support cancelation of an automation, or if an error occurs preventing successful cancelation, the service provider should respond to the PUT request with an HTTP status code 500 and an OSLC Error Resource (http://open-services.net/bin/view/Main/OslcCoreSpecification/Error_Responses) detailing the cause for the failed cancelation.

### State consistency

The Automation Request and Automation Result resources have an oslc:auto:state attribute. Automation service providers should, where possible, enforce state consistency for related Automation Requests and Results. In general, this means an Automation Result in a final state (completed, canceled) should not have a related Automation Request in a non-final state. Other contradictions such as completed Automation Result with a new Automation Request should also be avoided. Suggested consistent (C) and inconsistent (I) states are:

<table>
<thead>
<tr>
<th>Automation Result</th>
<th>new</th>
<th>queued</th>
<th>inProgress</th>
<th>canceling</th>
<th>canceled</th>
<th>complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoRequest</td>
<td>C I</td>
<td>I</td>
<td>I I I I I</td>
<td>I I</td>
<td>I I I I I</td>
<td>C C C</td>
</tr>
<tr>
<td></td>
<td>C C</td>
<td>C C C</td>
<td>C C C C C</td>
<td>C C C C C</td>
<td>C C C C C</td>
<td>C C C C</td>
</tr>
<tr>
<td></td>
<td>I I</td>
<td>I I I</td>
<td>C C C I I I</td>
<td>I I I I</td>
<td>I I I I</td>
<td>C C C</td>
</tr>
<tr>
<td></td>
<td>C C</td>
<td>C C C</td>
<td>C C C C C</td>
<td>C C C C C</td>
<td>C C C C C</td>
<td>C C C</td>
</tr>
</tbody>
</table>

### Parameters Added During Execution

When Automation Requests are created for an Automation Plan, the creator of the request supplies oslc:auto:inputParameter attributes based on the oslc:auto:parameterDefinition attributes found in the Automation Plan instance. There are scenarios where a provider may add additional parameters during the course of execution and a consumer of Automation Results might wish to discover what these added parameters will be. One method of discovery is for the consumer to simply examine the oslc:auto:outputParameter attributes of the Automation Result. This may not be sufficient for consumers who have a need to know the added parameters prior to executing the Automation Plan.

Service providers can advertise which parameters will be added during the course of execution using the oslc:readOnly attribute of the oslc:Property resource which is the basis for the oslc:auto:parameterDefinition in the Automation Plan. By setting oslc:readOnly to true, the provider indicates that this parameter is not available for the consumer to set, but will or may be added to the Automation Result's oslc:auto:outputParameters.

Whether it is guaranteed to be added to the Result is based on the value of oslc:occurs for the specific parameterDefinition.

**Example 1:** An Automation Plan parameterDefinition fragment showing a parameter guaranteed to be added during execution

```xml
<oslc_auto:parameterDefinition>
  <oslc:name>DeployedIPAddress</oslc:name>
  <oslc:readOnly>true</oslc:readOnly>
  <oslc:occurs>http://open-services.net/ns/core#Exactly-one</oslc:occurs>
  <oslc:valueType rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</oslc_auto:parameterDefinition>
```

**Example 2:** An Automation Plan parameterDefinition fragment showing a parameter which may be added during execution

```xml
<oslc_auto:parameterDefinition>
  <oslc:name>FailedTestName</oslc:name>
  <oslc:readOnly>true</oslc:readOnly>
  <oslc:occurs>http://open-services.net/ns/core#Zero-or-many</oslc:occurs>
  <oslc:valueType rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</oslc_auto:parameterDefinition>
```

### Appendix A: Samples

This section is non-normative, i.e. it does not affect compliance.

See OSLC Automation Version 2.1 Samples (http://open-services.net/wiki/automation/OSLC-Automation-Version-2.1-Samples)

### Appendix B: Resource Shapes

This section is non-normative, i.e. it does not affect compliance.


Arwe: At this point, the Resource Shapes link target does not exist. Need to decide which shapes we want, if any. Starting point might be one for each resource definition table, and one for each interaction pattern. Since they’re non-normative, we can defer until finalization if we want.

### Appendix C: Notices and References

License and Intellectual Property

We make this specification available under the terms and conditions set forth in the site Terms of Use (http://open-services.net/terms/), IP Policy (http://open-services.net/ip-policy/), and the Workgroup Participation Agreement for this Workgroup (http://open-services.net/legal-agreements/performance-monitoring-wpa/).

Reporting Issues on the Specification

The working group participants who author and maintain this working draft specification, monitor a distribution list where issues or questions can be raised, see Automation Mailing List (http://open-services.net/mailman/listinfo/oslc-automation-open-services.net).

Also the issues found with this specification and their resolution can be found at Automation Specification Version 2.0 Issues (http://open-services.net/wiki/automation/Automation-Specification-Version-2.0-Issues).

### START NEW FOR 2.1

### Appendix D: Changes

**Specification**

- (2.1 convergence issue 4) Added Execution environments section to clarify use of the property.
- (2.1 convergence issue 3) Clarified wording of oslc:futureAction description and “OSLC Actions and Automation / Discovering actions that will be executable after an Automation Request completes” section, also clarified “Resource: Dialog” table.
- (2.1 convergence) moved futureAction and executes predicates from Automation to Core vocabulary
- Core Actions section (wiki/core/Exposing-arbitrary-actions-on-RDF-resources/Future-actions) Auto Plan section
- Immediate-Execution Creation Dialog term added - gives formal name to 2.0 behavior, to differentiate from deferred execution cases
- Deferred-Execution new concept, to address scheduling scenarios
- Actions extensions to new Core concept, to cover teardown and scheduling scenarios
- (2.0 after-finalization rollout) The definition of finished was updated to match the 2.0 vocabulary.
- (2.0 after-finalization rollout) parameter definition examples fixed

**Vocabulary changes**

- Current live vocabulary via namespace redirect (/ns/auto), as RDF Schema File:auto.rdf (http://open-services.net/wiki/automation/File%3Aauto.rdf), and as HTML File:auto.html (http://open-services.net/wiki/automation/File%3Aauto.html) (which should be the same page served via the namespace redirect)

### New terms
**Contributors and Contact Information**

- John Arwa ([open-services.net/forums/member/149](http://open-services.net/forums/member/149))
- Umberto Caselli ([open-services.net/forums/member/414](http://open-services.net/forums/member/414))
- Michael Feldieg ([open-services.net/forums/member/265](http://open-services.net/forums/member/265))
- Paul McMahon ([open-services.net/forums/member/272](http://open-services.net/forums/member/272))
- Martin Pain ([open-services.net/forums/member/443](http://open-services.net/forums/member/443))
- Charles Rankin ([open-services.net/forums/member/281](http://open-services.net/forums/member/281))
- Stephen Rowles ([open-services.net/forums/member/574](http://open-services.net/forums/member/574))
- Steve Speicher ([open-services.net/forums/member/6](http://open-services.net/forums/member/6))

**Intellectual Property Covenant**

The members of the Working Group (or as appropriate, their employers) have documented a Patent Non-Assertion Covenant for implementations of the Automation 2.0 Specification, as described in the open-services.net Terms of Use ([open-services.net/html/Terms.html](http://open-services.net/html/Terms.html)). Details of the Covenant may be found [here](http://open-services.net/wiki/automation/Patent-Non-Assert-Covenants-for-Automation-Specification-version-2.0).

**References**

- OSLC Core - [OSLC Core Specification 2.0](http://open-services.net/bin/view/Main/OslcCoreSpecification)
- Dublin Core 1.1 - [Dublin Core Metadata Element Set, Version 1.1](http://dublincore.org/documents/2010/10/11/dces/)
- FOAF - [Friend of a Friend (FOAF) v0.98](http://xmlns.com/foaf/spec/20100809.html)
- HTTP 1.1 - [Hyper-text Transfer Protocol (HTTP/1.1)](http://tools.ietf.org/html/rfc2616)
- OAuth 1.0a - [RFC5849 - The OAuth 1.0 Protocol](http://tools.ietf.org/html/rfc5849)
- URI Syntax - [URI Generic Syntax](http://tools.ietf.org/html/rfc3986)
- XML Namespaces - [Namespaces in XML 1.0 (Third Edition)](http://www.w3.org/TR/REC-xml-names)
- XSD Datatypes - [XML Schema Part 2: Datatypes Second Edition](http://www.w3.org/TR/xmlschema-2)

All content [Creative Commons Attribution 3.0 US](http://creativecommons.org/licenses/by/3.0/us/) unless otherwise specified. See more [terms of use](/terms/).