Open Services for Lifecycle Collaboration
Tracked Resource Set Specification Version 2.0

Status: Finalizing Draft

This Version

- http://open-services.net/wiki/core/TrackedResourceSet-2.0/ (http://open-services.net/wiki/core/TrackedResourceSet-2.0/)

Latest Version

- http://open-services.net/wiki/core/TrackedResourceSet-2.0 (http://open-services.net/wiki/core/TrackedResourceSet-2.0)

Previous Version

- This is the first version of this specification.

Authors

- Steve Speicher (http://open-services.net/bin/view/Main/SteveSpeicher)
- Frank Budinsky
- Vivek Garg

Contributors

- See Contributors section below

Table of Contents

Contents

Notation and Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119 (http://www.ietf.org/rfc/rfc2119.txt). Domain name examples use RFC2606 (http://tools.ietf.org/html/rfc2606).

Introduction

The Tracked Resource Set protocol allows a server to expose a set of resources in a way that allows clients to discover the exact set of resources in the set, to track all additions to and removals from the set, and to track state changes to all resources in the set. The protocol does not assume that clients will dereference the resources. The protocol is suitable for dealing with large sets containing a large number of resources, as well as highly active resource sets that undergo continual change. The protocol is HTTP-based and follows RESTful principles.

Terminology

Resource Set - an enumerable, finite, collection of Resources

Resource - web resource identified by URI; the Resource Set members

Server - party playing the role of Resource Set provider

Client - party playing the role of consumer; interacts with a Server to enumerate and track Resources in the Server's Resource Set

Tracked Resource Set (TRS) - describes the set of Resources in a Resource Set, expressed as a Base and a Change Log

Base - portion of a Tracked Resource Set representation that lists member Resources

Change Log - portion of a Tracked Resource Set representation detailing a series of Change Events

Change Event - describes the addition, removal, or state change of a member Resource

Overview

The Server maintains a Resource Set. A Resource Set consists of a finite, enumerable set of Resources. Each Resource is identified by a URI. The Server will have its own criteria for determining the exact set of member Resources at any point in time. However, clients need not be aware of the Server's criteria, and will instead discover a Resource Set's members by interacting with the Server using the Tracked Resource Set protocol.

The Server **MUST** provide an HTTP(S) URI corresponding to its Resource Set. This is referred to as the Tracked Resource Set URI.

A HTTP GET request sent to the Tracked Resource Set URI returns a representation of the state of the Resource Set characterized in terms of a Base and a Change Log: the Base provides a point-in-time enumeration of the members of the Resource Set, and the Change Log provides a time series of adjustments describing changes to members of the Resource Set. When the Base is empty, the Change Log describes a history of how the Resource Set has grown and evolved since its inception. When the Change Log is empty, the Base is a simple enumeration of the Resources in the Resource Set. This hybrid base+delta form gives the Server flexibility to structure the representation in ways that are most useful to its Clients.

The Base portion of a Tracked Resource Set representation is an Linked Data Platform (LDP) Container where each member references a Resource that was in the Resource Set at the time the Base was computed. The Change Log portion is represented as multiple same-subject and same-predicate triples, where the objects correspond to Change Events. The order information is indicated within the Change Event entry itself. There **MUST NOT** be a gap between the Base portion and the Change Log portion of a Tracked Resource Set representation; however, the Change Log portion may contain earlier Change Event entries that would be accounted for by the Base portion. A "cutoff" property of the Base identifies the point in the Change Log at which processing of Change Events can be cut off because older changes are already covered by the Base portion.

Tracked Resource Set

An HTTP GET on a Tracked Resource Set URI returns a representation structured as follows (note: for exposition, the example snippets show the RDF information content using Turtle; the actual representation of these resources "on the wire" may vary):

```
# Resource: http://cm1.example.com/trackedResourceSet
@prefix trs: <http://open-services.net/ns/core/trs#> .

<http://cm1.example.com/trackedResourceSet>
  a trs:TrackedResourceSet ;
  trs:base <http://cm1.example.com/baseResources/> ;
  trs:changeLog [
    a trs:ChangeLog ;
    trs:change  ...  .
  ] .
```

A Tracked Resource Set **MUST** provide references to the Base and Change Log using the `trs:base` and `trs:changeLog` predicates respectively.

A typical Client will periodically poll the Tracked Resource Set looking for recent Change Events. In order to cater to this usage, the Tracked Resource Set's representation **MUST** contain the triples for the referenced Change Log (i.e., via a Blank Node, or an inline named Resource). Specifically the Tracked Resource Set representation will contain a triple `{TRS-URL, rdf:type, trs:TrackedResourceSet}` including the triples for the Change Events themselves enumerated in `{TRS-URL, trs:change, ChangeEvent-URI}` where the Change Events **MUST** be present in the Tracked Resource Set's representation. The Server **SHOULD** also support etags, caching, and conditional GETs for Tracked Resource Set resources and relegate the Base to separate resources.

Change Log

A Change Log provides a set of changes, the ordering of the changes is included with each change event. The following example illustrates the contents of a Change Log:

```
# Resource: http://cm1.example.com/trackedResourceSet
@prefix trs: <http://open-services.net/ns/core/trs#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://cm1.example.com/trackedResourceSet>
  a trs:TrackedResourceSet ;
  trs:base <http://cm1.example.com/baseResources/> ;
  trs:changeLog [
    a trs:ChangeLog ;
    trs:change <urn:urn-3:cm1.example.com:2010-10-27T17:39:33.000Z:103> ;
    trs:change <urn:urn-3:cm1.example.com:2010-10-27T17:39:32.000Z:102> ;
    trs:change <urn:urn-3:cm1.example.com:2010-10-27T17:39:31.000Z:101> .
  ] .

<urn:urn-3:cm1.example.com:2010-10-27T17:39:33.000Z:103>
  a trs:Creation ;
  trs:changed <http://cm1.example.com/bugs/23> ;
  trs:order "103"^^xsd:integer .

<urn:urn-3:cm1.example.com:2010-10-27T17:39:32.000Z:102>
  a trs:Modification ;
  trs:changed <http://cm1.example.com/bugs/22> ;
  trs:order "102"^^xsd:integer .

<urn:urn-3:cm1.example.com:2010-10-27T17:39:31.000Z:101>
  a trs:Deletion ;
  trs:changed <http://cm1.example.com/bugs/21> ;
  trs:order "101"^^xsd:integer .
```

As shown, a Change Log provides a set of Change Event entries in a multi-valued RDF property called `trs:change`.

Change Events **MUST** have URIs (i.e., they cannot be Blank Nodes) to allow Clients to recognize entries they have seen before. The URI is only used to identify an event (i.e., it need not be HTTP GETable) and therefore MAY be a URN, as shown in the example.

Each Change Event has a sequence number, `trs:order`; sequence numbers are non-negative integer values that increase over time. A Change Event entry carries the URI of the changed Resource, trs:changed, and an indication, via `rdf:type` (a.k.a. “a” in Turtle), of whether the Resource was added to the Resource Set, removed from the Resource Set, or changed state while a member of the Resource Set. The entry with the highest `trs:order` value (i.e., 103 in this example) is the most recent change. As changes continue to occur, a Server **MUST** add new Change Events to the newest Change Log segment. The sequence number (i.e., `trs:order`) of newer entries **MUST** be greater than previous ones. The sequence numbers MAY be consecutive numbers but need not be.

Note that the actual time of change is not included in a Change Event. Only a sequence number, representing the “sequence in time” of each change is provided. The URI of a Change Event **MUST** be guaranteed unique, even in the wake of a Server rollback where sequence numbers get reused. A time stamp MAY be used to generate such a URI, as in the above example, although other ways of generating a unique URI are also possible.

A Change Log represents a series of changes to its corresponding Resource Set over some period of time. The Change Log **MUST** contain Change Events for every Resource creation, deletion, and modification during that period. A Server **MUST** report a Resource modification event if a GET on it would return a semantically different response from previously. For a resource with RDF content, a modification is anything that would affect the set of RDF triples in a significant way. A Server **MAY** safely report a modification event even in cases where there would be no significant difference in response. Some cases of modifications that would be considered semantically different from previous or significant difference would be: inserted triple, removed triple, triple replaced (new object/literal, e.g. changing boolean literal “true” to “false”), replaced vocabulary term used (e.g. change from `dcterms:title` to `rdfs:label`).

The Server **SHOULD NOT** report unnecessary Change Events although it might happen, for example, if changes occur while the base is being computed. A Client **SHOULD** ignore a creation event for a Resource that is already a member of the Resource Set, and **SHOULD** ignore a deletion or modification event for a Resource that is not a member of the Resource Set.

Change Log Segmentation

The Change Log in the previous example consisted of a single `trs:ChangeLog` resource. Typically, however, the Change Log will be very large, requiring the changes to be segmented into multiple smaller `trs:ChangeLog` resources:

```
# Resource: http://cm1.example.com/trackedResourceSet
@prefix trs: <http://open-services.net/ns/core/trs#> .

<http://cm1.example.com/trackedResourceSet>
  a trs:TrackedResourceSet ;
  trs:base <http://cm1.example.com/baseResources/> ;
  trs:changeLog [
    a trs:ChangeLog ;
    trs:change <urn:urn-3:cm1.example.com:2010-10-27T17:39:33.000Z:103> ;
    trs:change <urn:urn-3:cm1.example.com:2010-10-27T17:39:32.000Z:102> ;
    trs:change <urn:urn-3:cm1.example.com:2010-10-27T17:39:31.000Z:101> ;
    trs:previous <http://cm1.example.com/changeLog/1> .
] .

<urn:urn-3:cm1.example.com:2010-10-27T17:39:33.000Z:103>
...
```

and then…

```
# Resource: http://cm1.example.com/changeLog/1
@prefix trs: <http://open-services.net/ns/core/trs#> .

<http://cm1.example.com/changeLog/1>
  a trs:ChangeLog ;
  trs:change <urn:urn-3:cm1.example.com:2010-10-27T17:39:30.000Z:100>, {more stuff} .

<urn:urn-3:cm1.example.com:2010-10-27T17:39:30.000Z:100>
...
```

As shown, the `trs:previous` reference is used in this case to connect to the Change Log resource containing the next group of chronologically earlier Change Events. The most recent Change Events are included in the Tracked Resource Set itself. This allows a Client to easily discover the most recent Change Event, and retrieve successively older Change Log resources until it encounters a Change Event that has already been processed (on a previous check). The protocol does not attach significance to where a Server breaks the Change Log into separate parts, i.e., the number of entries in a trs:ChangeLog is entirely up to the Server.

To allow Clients to retrieve the Change Events in a Change Log segment using a single HTTP GET request, Servers **MUST** include all of the triples corresponding to a Change Log segment in the same HTTP response (i.e., in the representation of either the Tracked Resource Set or a trs:previous Change Log). This includes triples whose subject is the Change Log, the `trs:change` entries, and the Change Events themselves. Other than the Change Events, all of these **MAY** be represented using Blank Nodes.

Truncated Change Logs

A chain of Change Logs **MAY** continue all the way back to the inception of the Resource Set and contain Change Events for every change made since then. However, to avoid maintaining this ever growing list of Change Logs indefinitely, a Server **MAY** truncate the log at a suitable point in the chain. This can be accomplished by deleting the oldest segments of the Change Log and/or by removing the triples that reference them. In any case, Clients **MUST** be prepared to receive HTTP status code 404 (Not found) when navigating the “previous” reference from a final or stale Change Log segment.

To ensure that a new Client can always get started, the Change Log **MUST** contain the base cutoff event of the corresponding Base, and all Change Events more recent than it. Thus the Server is only allowed to truncate Change Events older than the base cutoff event. When the Base has no base cutoff event (i.e., the Base enumerates the Resource Set at the start of time), the Change Log **MUST** contain all Change Events back to the start of time; i.e., no truncation is allowed.

To minimize the likelihood of Clients falling too far behind and losing information, it is **STRONGLY RECOMMENDED** that a Server retain a minimum of seven days worth of Change Events.

Base Resources

The Base of a Tracked Resource Set is a W3C Linked Data Platform (LDP) Container where each member references a Resource that was in the Resource Set at the time the Base was computed. HTTP GET on a Base URI returns an LDP Container with the following structure:

```
# Resource: http://cm1.example.com/baseResources/
@prefix trs: <http://open-services.net/ns/core/trs#> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .

<http://cm1.example.com/baseResources/>
  a ldp:DirectContainer;
  ldp:membershipResource <http://cm1.example.com/baseResources/>;
  ldp:hasMemberRelation ldp:member;
  trs:cutoffEvent <urn:urn-3:cm1.example.com:2010-10-27T17:39:31.000Z:101> ;
  ldp:member <http://cm1.example.com/bugs/1> ;
  ldp:member <http://cm1.example.com/bugs/2> ;
  ldp:member <http://cm1.example.com/bugs/3> ;
  ...
  ldp:member <http://cm1.example.com/bugs/199> ;
  ldp:member <http://cm1.example.com/bugs/200> .
```

Each Resource in the Resource Set **MUST** be referenced from the container using an LDP membership predicate.

Because of the highly dynamic nature of the Resource Set, a Server may have difficulty enumerating the exact set of resources at a point in time. Because of that, the Base can be only an approximation of the Resource Set. A Base might omit mention of a Resource that ought to have been included or include a Resource that ought to have been omitted. For each erroneously reported Resource in the Base, the Server **MUST** at some point include a corrective Change Event in the Change Log more recent that the base cutoff event. The corrective Change Event corrects the picture for that Resource, allowing the Client to compute the correct set of member Resources. A corrective Change Event might not appear in the Change Log that was retrieved when the Client dereferenced the Tracked Resource Set URI. The Client might only see a corrective Change Event when it processes the Change Log resource obtained by dereferencing the Tracked Resource Set URI on later occasions.

A Server **MUST** refer to a given resource using the exact same URI in the Base (membership triple) and every Change Event (`trs:changed` reference) for that resource.

The response representation of a Base **MUST** include a `trs:cutoffEvent` property, whose value is the URI of the most recent Change Event in the corresponding Change Log that is already reflected in the Base. This corresponds to the latest point in the Change Log from which a Client can begin incremental monitoring/updating if it wants to remain synchronized with further changes to the Resource Set. As mentioned above, the cutoff Change Event **MUST** appear in the non-truncated portion of the Change Log. When the `trs:cutoffEvent` is `rdf:nil`, the Base enumerates the (possibly empty) Resource Set at the beginning of time.

Paged Base

Note (Feature Unstable): The paging support is based on the W3C Linked Data Platform (Paging) Specification (https://dvcs.w3.org/hg/ldpwg/raw-file/default/ldp-paging.html) that has not stabilized.

The Base **MAY** be broken into multiple pages in which case the Server will respond with a 30x redirect message, directing the Client to the first "single-page resource". The representation of a single-page resource will contain a subset of the Base's membership triples. In addition, it will contain response header indicating a reference to the next page.

Below is an example of server-initiated paging and response headers:

HTTP Request:

```
GET /baseResources/ HTTP/1.1
Host: cm1.example.com
Accept: text/turtle
```

HTTP Response:

```
HTTP/1.1 303 See Other
Location: http://cm1.example.com/baseResources/page1
```

Following the redirect (server-initiated paging):

HTTP Request:

```
GET /baseResources/page1 HTTP/1.1
Host: cm1.example.com
Accept: text/turtle
```

HTTP Response:

```
HTTP/1.1 200 OK
Content-Type: text/turtle
Date: Wed, 11 Jun 2014 12:55:05 GMT
ETag: 2014-06-10T14:05:44.18Z
Link: <http://cm1.example.com/baseResources/page1>; rel="first",
      <http://cm1.example.com/baseResources/page2>; rel="next",
      <http://www.w3.org/ns/ldp#Page>; rel="type"

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

<http://cm1.example.com/baseResources/>
  a ldp:DirectContainer;
  ldp:membershipResource <http://cm1.example.com/baseResources/>;
  ldp:hasMemberRelation ldp:member;
  trs:cutoffEvent <urn:urn-3:cm1.example.com:2010-10-27T17:39:31.000Z:101> ;
  ldp:member <http://cm1.example.com/bugs/1> ;
  ldp:member <http://cm1.example.com/bugs/2> ;
  ldp:member <http://cm1.example.com/bugs/3> .
```

The last page in the list is indicated by omitting the link relation for next, for example omitting `Link: rel="next"`. The Tracked Resource Set protocol does not attach significance to the order in which a Server enumerates the resources in the Base or breaks the Base up into pages.

The first single-page resource of a Base **MUST** include a `trs:cutoffEvent` property.

When a Base is broken into pages, the Client will discover and retrieve Base page resources to determine the Resources in the Base. A Client **MUST** retrieve all the page resources of the Base to compute the complete set of resources in the Base. A Client **MAY** retrieve the Base page resources in any order, including retrieving some Base page resources in parallel. A Client retrieves the Base page resources at its own pace, and **MAY** retrieve any of the Base page resources more than once. If the Server allows the representation of Base page resources to vary over time, the Server **MUST** ensure that the set of Resources a Client would infer as members is necessarily an approximation of the Resource Set which, when corrected by Change Events after the Base's cutoff event, yields the correct set of member Resources in the Resource Set.

Resources

This section defines the resources of the Tracked Resource Set specification. TRS servers **MUST** support Turtle (i.e., text/turtle) representations of these resources. TRS servers **MAY** provide representations of the requested TRS resources beyond those necessary to conform to this specification, using standard HTTP content negotiation. If the client does not indicate a preference, text/turtle **MUST** be returned.

Tracked Resource Set Namespace

The namespace used for resources and properties defined in this specification is as follows:

- Namespace URI: `http://open-services.net/ns/core/trs#`
- Default Prefix: `trs`

Resource: TrackedResourceSet

- **Name:** `TrackedResourceSet`
- **Description:** A Tracked Resource Set provides a representation of the current state of a Resource Set.
- **Type URI:** `http://open-services.net/ns/core/trs#TrackedResourceSet`

TrackedResourceSet Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
trs:base	exactly-one	True	Resource	Reference	ldp:DirectContainer	An enumeration of the Resources in the Resource Set.
trs:changeLog	exactly-one	True	Resource	Either	trs:ChangeLog	A Change Log providing a time series of incremental adjustments to the Resource Set.

A Base resource (i.e., target of the `trs:base` predicate) has the following properties:

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
ldp:member	zero-or-many	True	Resource	Reference	any	A member Resource of the Resource Set. `ldp:member` is the preferred predicate, in either case it will be indicated by `ldp:hasMemberRelation`
trs:cutoffEvent	exactly-one	True	Resource	Either	trs:Creation, trs:Modification, trs:Deletion	The most recent Change Log entry that is accounted for in this Base. When `rdf:nil`, the Base is an enumeration at the start of time.

Resource: ChangeLog

- **Name:** `ChangeLog`
- **Description:** A Change Log describes what resources have been created, modified or deleted, and when.
- **Type URI:** `http://open-services.net/ns/core/trs#ChangeLog`

ChangeLog Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
trs:change	zero-or-many	True	Resource	Either	trs:Creation, trs:Modification, trs:Deletion	The Change Event entries.
trs:previous	zero-or-one	True	Resource	Either	trs:ChangeLog	The continuation of the Change Log, containing the next group of chronologically earlier Change Events.

Each object of a `trs:change` triple is a Resource representing a Change Event with the following properties:

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
rdf:type	exactly-one	True	Resource	Reference	any	The type of the Change Event. One of trs:Creation, trs:Modification, or trs:Deletion.
trs:changed	exactly-one	True	Resource	Either	any	The Resource that has changed.
trs:order	exactly-one	True	Integer (Non-negative)	Either	n/a	The sequence in time of the Change Event.

Client Behavior

(this section is informative)

This section describes one (relatively straightforward) way that a Client can use the Tracked Resource Set protocol to build and maintain its own local replica of a Server's Resource Set.

Initialization procedure

A Client wishing to determine the complete collection of Resources in a Server's Resource Set, so that it can build its local replica of the Resource Set, proceeds as follows:

- Send a GET request to the Tracked Resource Set URI to retrieve the Tracked Resource Set representation to learn the URI of the Base.
- Use GET to retrieve successive pages of the Base, adding each of the member Resources to the Client's local replica of the Resource Set.
- Invoke the Incremental Update procedure (below). The sync point event is the trs:cutoffEvent property (on the first page of the Base). A clever Client might run this step in parallel with the previous one in an effort to prevent the case where the Client can't catch up to the current state of the Resource Set using the Change Log (after initial processing) because initial processing takes too long.

The overall work to build the local replica of the Resource Set is linear in the size of the Base plus the number of Change Events that occurred after the base cutoff event. The Server can help Clients building new local replicas of its Resource Set by providing as recent a Base as possible, because that means the Client will have to process fewer Change Events. It is entirely up to the Server how often it computes a new Base. It is also up to the Server how it computes the members of a Base, whether by enumerating its Resource Set directly (e.g., by querying an underlying database), or perhaps by coalescing its internal change log entries into a previous base.

Incremental Update procedure

Suppose now that a Client has a local replica of the Server's Resource Set that is accurate as of a particular sync point event known to the Client. A Client wishing to update its local replica of the Server's Resource Set acts as follows:

- Send a GET request to the Tracked Resource Set URI to retrieve the Tracked Resource Set representation to learn its current Change Log.
- Search through the chain of Change Logs from newest to oldest to find the sync point event. The incremental update fails if the Client is unable to locate the sync point (i.e., it gets to the end of the log).
- Process all Change Events after the sync point event, from oldest to newest, making corresponding changes to the Client's local replica of the Resource Set. Record the latest event processed as the new sync point event. A clever Client might record (some number of) recently processed events for possible future undo in the event of a server rollback.

When the procedure succeeds, the Client will have updated its own local replica of the Server's Resource Set to be an accurate reflection of the set of resources as described by the retrieved representation of the Tracked Resource Set. Of course, the Server's actual Resource Set may have undergone additional changes since then. While the Client may never catch up to the Server, it can at least keep its local replica of the Resource Set almost up to date. By choosing the interval at which it polls for updates, a Client controls how long the two are allowed to drift apart. The overall work to maintain the local replica of the Resource Set is linear in the length of the Change Event stream. In the (hopefully rare) situation that the Client fails to find its sync point event, one of two things is likely to have happened on the Server: either the Server has truncated its Change Log, or the Server has been rolled back to an earlier state.

If the Client had been retaining a local record of previously processed events, the Client may be able to detect a Server rollback if it notices the successor event of some previously processed event has been removed or changed to one with a different identifier than before. In this case, the Client can undo changes to its local replica back to that sync point, and then pick up processing from there.

Once the Incremental Update procedure fails, it is unlikely to succeed in the future. The Client has reached an impasse. The Client's only way forward is to discard its local replica and start over.

Discoverability

The TRS servers MUST document the URLs of its Tracked Resource Set resources in the their documentation.

In order to help an administrator of a TRS client-based application in configuring its TRS server endpoints, a TRS server MAY also make its Tracked Resource Sets discoverable. Discoverability is a convenience; an administrator can configure a TRS client with a particular Tracked Resource Set knowing just its URL.

An application declares the existence and location of a Tracked Resource Set with a statement of the following form:

```
@prefix trs: <http://open-services.net/ns/core/trs#> .

<http://cm1.example.com/app/> trs:trackedResourceSet <trackedResourceSet>.
```

Applications **MAY** provide multiple Tracked Resource Sets.

Appendix A: Samples

(this section is informative)

See samples within the body of this specificiation.

Appendix B: Resource Shapes

Not applicable

Appendix C: Notices and References

Contributors

- SteveSpeicher (http://open-services.net/bin/view/Main/SteveSpeicher) (IBM, OSLC-Core Lead)
- Frank Budinsky (IBM, OSLC-Core)
- Vivek Garg (IBM, OSLC-Core)
- Arthur Ryman (IBM, OSLC-Core)
- Nick Crossley (IBM, OSLC-Core)
- Ian Green (IBM, OSLC-Core)

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 Core Mailing List (http://open-services.net/mailman/listinfo/oslc-core_open-services.net)

Also the issues found with this specification and their resolution can be found at Core 2.0 Issues (http://open-services.net/wiki/core/OSLC-Core-V2-Issues/).

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/).

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)
- JSON - JavaScript Object Notation (http://json.org/)
- OAuth 1.0a - RFC5849 - The OAuth 1.0 Protocol (http://tools.ietf.org/html/rfc5849)
- RDF/XML Concepts - RDF/XML Concepts and Abstract Syntax (http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/)
- RDF/XML Syntax - RDF / XML Syntax Specification (Revised) (http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/)
- 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)
- Linked Data Platform - Linked Data Platform 1.0 (http://www.w3.org/TR/ldp/)

Appendix D: Changes

- Changed property `trs:changes` to `trs:change` (dropped the s)

Category:Specifications (http://open-services.net/wiki/core/Category:Specifications)

Categories

- Specifications (http://open-services.net/wiki/core/Category:Specifications)
