CHARMe node Interface Control Document

CHARMe is funded by the EC under its FP7 Research Programme

Document Control

Contributors

Person	Role	Organisation	Contribution
M.Nagni	Developer	STFC	Initial Draft.
A.Wilson	Developer	STFC	Update

Document Approval

Person	Role	Organisation

References

ID	Author	Document Title	Date
[R-1]			

Revision History

Issue	Author	Date	Description
0.1	M.Nagni	16th Aug 2013	Initial Draft.
0.2	M.Nagni	26th Sept 2013	Extended examples. Preliminary integration with OpenSearch.
0.3	A.Wilson	09th Jan 2015	Version 0.7.0 of the code.

Namespaces Used

Prefix	Namespace	Description
cito	http://purl.org/spar/cito/	Citation Typing Ontology
cnt	http://www.w3.org/2011/content#	Representing Content in RDF
dc	http://purl.org/dc/elements/1.1/	Dublin Core
dcterms	http://purl.org/dc/terms/	Dublin Core Terms

dctypes	http://purl.org/dc/dcmitype/	Dublin Core Metadata Initiative
fabio	http://purl.org/spar/fabio/	FRBR-aligned Bibliographic Ontology
foaf	http://xmlns.com/foaf/0.1/	The Friend of a Friend namespace
oa	http://www.w3.org/ns/oa#	The Open Annotation ontology
prov	http://www.w3.org/ns/prov#	The PROV namespace
skos	http://www.w3.org/2004/02/skos/core#	Simple Knowledge Organization System Namespace Document
rdf	http://www.w3.org/1999/02/22/22-rdf-syntax-ns#	The RDF namespace
xsd	http://www.w3.org/2001/XMLSchema#	XML Schema namespace

Table of Contents

List of Figures

1 Node Services

- *advance_status*

  Action: Changes an annotation state
  HTTP Method: POST
  Content-Type: [application/ld+json]
  Parameters:
  - *annotation* – annotation_id
  - *toState* – ['submitted'|'stable'|'invalid'|'retired']

  Returns a HTTP code

- *data/(w+)?(format=[xml,turtle,json-ld], depth=[int])*

  Action: Returns an RDF description of the node resource associated with a given ID
  HTTP Method: GET
  HTTP Accept: [application/rdf+xml, text/turtle, application/ld+json]
  Parameters:
  - *format* – the mime format of the HTTP Response. This parameter overrides any HTTP Accept
  - *depth* – an integer greater or equal to zero. Limit the details in the triple description

- *endpoint*

  Implements the SPARQL 1.1 HTTP Graph Store specifications.

- *index/[submitted|invalid|stable|retired]?(format=[xml,turtle,json-ld])*

  Action: Returns the annotation nodes in the specific named graph
  HTTP Method: GET
  Parameters:
  - *format* – the mime format of the HTTP Response. This parameter overrides any HTTP Accept

- *insert/annotation*

  Action: Inserts a RDF complaint document in the *submitted* graph. Provenance data about the user is added to the annotation.
  HTTP Method: POST
  Content-Type: [application/rdf+xml, text/turtle, application/ld+json]
  Accept: [text/plain]
  Returns the URI of the new annotation

- *modify/annotation*

  Action: Move the current annotation to *retired* graph and then insert the RDF complaint document in the *submitted* graph. Provenance data about the user is added to the annotation and links are created between the submitted and retired annotations.
  HTTP Method: POST
  Content-Type: [application/rdf+xml, text/turtle, application/ld+json]

Accept: [text/plain]
Returns the URI of the new annotation

- *page/(w+)?(depth=[int])*
  Action: Returns an RDF description of the node resource associated with a given ID
  HTTP Method: GET
  HTTP Accept: [text/html]
  Parameters:
    - *depth* – an integer greater or equal to zero. Limit the details in the triple description

- *resource/(w+)/reporttomoderator/*
  Action: Email the registered admin for the site where the annotation originated. Include the OPTIONAL content of the PUT in the email. This should be used to explain why this annotation is being reported.
  HTTP Method: PUT
  HTTP Accept: [text/html]

- *resource/(w+)?(format=[xml,turtle,json-ld], depth=[int])*
  Action: Returns an RDF description of the node resource associated with a given ID
  HTTP Method: GET
  HTTP Accept: [application/rdf+xml, text/turtle, application/ld+json, text/html]
  Parameters:
    - *format* – the mime format of the HTTP Response. This parameter overrides any HTTP Accept
    - *depth* – an integer greater or equal to zero. Limit the details in the triple description

- *resource/(w+)*
  Action: Moves the resource to the 'retired' graph
  HTTP Method: DELETE

- *search/description*
  Action: Returns an XML document describing the implemented OpenSearch
  HTTP Method: GET

- *search/[atom]?count={count?}&startPage={startPage?}&startIndex={startIndex?}&bodyType={a0:type?}&citingType={a0:type?}&comment={a1:chars?}&dataType={a0:type?}&domainOfInterest={a2:domainOfInterest?}&motivation={a2:motivation?}&organization={a3:name?}&status={a2:status?}&target={a2:target?}&title={a4:title?}&userName={a3:accountName?}&depth={a2:depth?}&format={a2:format?}*
  Action: Searches inside the Annotations
  HTTP Method: GET
  Parameters:
    - *count* – the number of search results per page desired by the search client, default 10

- *startIndex* – the index of the first search result desired by the search client
- *startPage* – the page number of the set of search results desired by the search client
- *bodyType* – a URI of a rdf:type of an annotation body
- *citingType* – a URI of a rdf:type of an object citing a target
- *comment* – text to search inside the *cnt:chars* element of a body entity
- *dataType* – a URI of a rdf:type of an annotation target
- *domainOfInterest* – a URI of a domain of interest of an annotation
- *motivation* – a URI of a motivation of an annotation
- *organization* – a URI of an organization from where a client created an annotation
- *target* – a URI of a target of an annotation
- *title* – text to search inside the *dcterms:title* element of a citing entity
- *userName* – the user name of the creator of an annotation
- *status* – ['submitted'|'stable'|'invalid'|'retired'], default stable
- *depth* – an integer greater or equal to zero. Limit the details in the triple description
- *format* – the format of the returned elements ['json-ld'|'turtle'|'xml']

- *suggest/[atom]?count={count?}&startPage={startPage?}&startIndex={startIndex?}&q={searchTerms?}&bodyType={a0:type?}&citingType={a0:type?}&comment={a1:chars?}&dataType={a0:type?}&domainOfInterest={a2:domainOfInterest?}&motivation={a2:motivation?}&organization={a3:name?}&status={a2:status?}&target={a2:target?}&title={a4:title?}&userName={a3:accountName?}&depth={a2:depth?}&format={a2:format?}*

  Action: Searches for the values of one or more of the searchTerms. These can be restricted by providing parameter values.
  HTTP Method: GET
  Parameters:
  - *count* – the number of search results per page desired by the search client, default 10
  - *startIndex* – the index of the first search result desired by the search client
  - *startPage* – the page number of the set of search results desired by the search client
  - *q* – a space separated list from: ['body*Type*'|'*citingType*'|'*dataType*'|'*domainOfInterest*'|'*motivation*'|'*organization*'|'*']
  - *bodyType* – a URI of a rdf:type of an annotation body
  - *citingType* – a URI of a rdf:type of an object citing a target
  - *comment* – text to search inside the *cnt:chars* element of a body entity
  - *dataType* – a URI of a rdf:type of an annotation target
  - *domainOfInterest* – a URI of a domain of interest of an annotation
  - *motivation* – a URI of a motivation of an annotation
  - *organization* – a URI of an organization from where a client created an annotation
  - *target* – a URI of a target of an annotation
  - *title* – text to search inside the *dcterms:title* element of a citing entity
  - *userName* – the user name of the creator of an annotation
  - *status* – ['submitted'|'stable'|'invalid'|'retired'], default stable

- *depth* – an integer greater or equal to zero. Limit the details in the triple description
- *format* – the format of the returned elements ['json-ld'|'turtle'|'xml']

- *sparql*

  Implements the SPARQL read only endpoint

- *sparql.html*

  The FUSEKI read only GUI

- *version*

  Action: Returns the version of the server
  HTTP Method: GET

- *vocab*

  Action: Returns an RDF description of the CHARMe vocabulary
  HTTP Method: GET
  HTTP Accept: [application/rdf+xml, text/turtle, application/ld+json, text/html]

2 Creating Annotations

A CHARMe annotation is based on the OpenAnnotation (OA) specifications. As consequence any client implementation is responsible for assembling the annotation according to the OA documentation.

2.1 Using AnnoId And BodyId To Create Annotations

The CHARMe node is responsible for creating URIs for annotations and resolving them to specific resources. To achieve this, the submitted annotation should make use of key words in the annotation that will be converted to URIs by the CHARMe node during the ingest process. An annotation written in Turtle should have the general format:

```
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .

<chnode:annoID> a oa:Annotation ;
        oa:hasTarget <anyTargetURI> ;
        oa:hasBody <chnode:bodyID> .

<chnode:bodyID> a someClass ;
        [body properties...] .
```

Figure 1: The use of annonID and bodyID

Where ***annoID*** is mandatory and ***bodyID*** is optional. The OA specification states that an annotation may contain multiple bodies, however when inserting a new annotation to the CHARMe node it should have zero or one body marked as bodyID. Additional bodies may be included provided they reference existing URIs, either internal or external to the node.

The ***http://localhost/*** reference will also be automatically replaced by the node with its own URL.

2.2 Provenance and Other Annotation Metadata

Clients MUST provide the motivation for the annotation (`oa:motivatedBy`), the serialization time stamp (`oa:serializedAt`) and information about the software agent that created or updated the annotation (`oa:serializedBy`). An annotation can either point to an existing agent or a client can create a new one. If creating a new agent it MUST include the type `prov:SoftwareAgent` and `foaf:name`. For example:

```
@prefix chnode: <http://localhost/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<chnode:annoID> a oa:Annotation ;
        oa:hasTarget <anyTargetURI> ;
        oa:motivatedBy oa:tagging ;
        oa:serializedAt "2014-06-13T23:29:49.729746"^^xsd:dateTime ;
        oa:serializedBy <chnode:agentID> .

<chnode:agentID> a prov:SoftwareAgent ;
        foaf:name "CURL" .
```

Figure 2: Software Agent Provenance Data

Where ***agentID*** is substituted by the node during ingest.

When a new annotation is submitted the node adds additional provenance data, which is based on the

token used for the insert. The node creates a new `foaf:Person` object which contains the user name (`foaf:accountName`), family name (`foaf:familyName`) and given name (`foaf:givenName`). The node also attaches a time stamp (`oa:annotatedAt`) to the annotation.

All URIs in the submitted annotation MUST be URL encoded, see percent-encoding.

2.3 Using TargetID To Create Composite Targets

The use of ***targetID***, which will get substituted by the node during ingest, will create a target resource. However its use is restricted to targets of type `oa:Composite` and `charme:DatasetSubset.`

```
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .

<chnode:annoID> a oa:Annotation ;
        oa:hasTarget <chnode:targetID> .

<chnode:targetID> a oa:Composite ;
        oa:item <http://example.edu/target/01> ;
        oa:item <http://example.edu/target/02> .
```

Figure 3: The use of targetID for Composite

2.4 Tags For Fine Grained Commentary

A number of tags have been introduced for use by fine grained commentary. The ***targetID*** should be used for the DatasetSubset. ***subsetSelectorID*** should be used for the subsetSelector. The following tags represent multivalued resources, ***spatialExtentID***, ***temporalExtentID***, ***variableID***, ***verticalExtentID***. Each of these tags may have a prefix in order to disinguish the same instance of a resource, i.e. spatialExtentID-01, spatialExtentID-02.

```
@prefix charme: <http://purl.org/voc/charme#> .
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .

<chnode:annoID> a oa:Annotation ;
        oa:hasTarget <chnode:targetID> .

<chnode:targetID> a charme:DatasetSubset ;
        oa:hasSelector <chnode:subsetSelectorID> .

<chnode:subsetSelectorID> a charme:SubsetSelector ;
        charme:hasSpatialExtent <chnode:spatialExtentID-01> ;
        charme:hasSpatialExtent <chnode:spatialExtentID-02> ;
        charme:hasVariable <chnode:variableID-01> ;
        charme:hasVariable <chnode:variableID-02> .

<chnode:spatialExtentID-01> a charme:SpatialExtent .

<chnode:spatialExtentID-02> a charme:SpatialExtent .

<chnode:variableID-01> a charme:Variable .

<chnode:variableID-02> a charme:Variable .
```

Figure 4: Fine Grained Commentary

3 OA Examples

3.1 Text-based Annotations

This is the simple case where a user writes a free text comment about a given target. The text is created and attached as a new annotation body. Unstructured plain text is stored as `cnt:ContentAsText`. Structured content is also possible via `cnt:ContentAsXML` for representing for example XHTML.

```
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix cnt: <http://www.w3.org/2011/content#> .
@prefix dctypes: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XML-Schema#> .

<chnode:annoID> a oa:Annotation ;
        oa:hasTarget <http://example.edu/target/01> ;
        oa:hasBody <chnode:bodyID> ;
        oa:motivatedBy oa:commenting ;
        oa:serializedAt "2014-06-13T23:29:49.729746"^^xsd:dateTime ;
        oa:serializedBy <chnode:agentID> .

<chnode:agentID> a prov:SoftwareAgent ;
        foaf:name "CURL" .

<http://example.edu/target/01> a dctypes:Text ;
        dc:format "html/text" .

<chnode:bodyID> a cnt:ContentAsText, dctypes:Text ;
        cnt:chars "hello there!" ;
        dc:format "text/plain" .
```

Figure 5: A Text Annotation

In the next example the body refers to an external resource consequently the client does not need a ***bodyID***.

```
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix cnt: <http://www.w3.org/2011/content#> .
@prefix dctypes: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<chnode:annoID> a oa:Annotation ;
       oa:hasTarget <http://example.edu/target/01> ;
       oa:hasBody <http:example.edu/onepage> ;
       oa:motivatedBy oa:commenting ;
       oa:serializedAt "2014-06-13T23:29:49.729746"^^xsd:dateTime ;
       oa:serializedBy <chnode:agentID> .

<chnode:agentID> a prov:SoftwareAgent ;
       foaf:name "CURL" .

<http://example.edu/target/01> a dctypes:Text ;
       dc:format "html/text" .

<http:example.edu/onepage> a dctypes:Text ;
       dc:format "html/text" .
```

Figure 6: An Annotation with an External Body

3.2 Multiple Bodies

This situation may happen when the user wants to refer multiple notes to a single target; in this case is assumed that the user wants to insert zero or one textual while using older/external references for the other bodies.

3.3 Empty Body

This situation may happen when the user wants to simply generate a “bookmark” and update it later.

3.4 Multiple Targets

The body is considered to be equally related to each target individually, rather than the complete set of targets. This construction may be used so long as dropping any of the targets would not invalidate the annotation's meaning[1].

1 http://www.openannotation.org/spec/core/core.html#MultipleBodyTarget

```
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix cnt: <http://www.w3.org/2011/content#> .
@prefix dctypes: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<chnode:annoID> a oa:Annotation ;
        oa:hasTarget <http://example.edu/target/01> ;
        oa:hasTarget <http://example.edu/target/02> ;
        oa:hasBody <chnode:bodyID> ;
        oa:motivatedBy oa:commenting ;
        oa:serializedAt "2014-06-13T23:29:49.729746"^^xsd:dateTime ;
        oa:serializedBy <chnode:agentID> .

<chnode:agentID> a prov:SoftwareAgent ;
        foaf:name "CURL" .

<http://example.edu/target/01> a dctypes:Text ;
        dc:format "html/text" .

<http://example.edu/target/02> a dctypes:Text ;
        dc:format "html/text" .

<chnode:bodyID> a cnt:ContentAsText, dctypes:Text ;
        cnt:chars "hello there!" ;
        dc:format "text/plain" .
```

Figure 7: An Annotation with Multiple Targets

3.5 Composite Targets

Composite is a multiplicity construct that conveys that all of the constituent resources are required for the annotation to be correctly interpreted. There MUST be two or more item relationships (`oa:item`) for each composite (`oa:Composite`).

In order to allow the creation of composite resources for a target, a target resource needs to be created. To create a composite resource as a target use ***`targetID`***, which will get substituted by the node during ingest.

```
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .

<chnode:annoID> a oa:Annotation ;
        oa:hasTarget <chnode:targetID> .

<chnode:targetID> a oa:Composite ;
        oa:item <http://example.edu/target/01> ;
        oa:item <http://example.edu/target/02> .
```

Figure 8: An Annotation with a Composite Target

3.6 Other Formats

Figures 5 and 6 show the JSON-LD and XML-RDF representations of the Turtle from figure 3.

```
json-ld
{
"@graph": [
  {
    "@id": "http://localhost/annoID",
    "@type": [
      "http://www.w3.org/ns/oa#Annotation"
    ],
    "http://www.w3.org/ns/oa#hasBody": [
      {
        "@id": "http://localhost/bodyID"
      }
    ],
    "http://www.w3.org/ns/oa#hasTarget": [
      {
        "@id": "http://example.edu/target/01"
      }
    ],
    "http://www.w3.org/ns/oa#motivatedBy": [
      {
        "@id": "http://www.w3.org/ns/oa#commenting"
      }
    ],
    "http://www.w3.org/ns/oa#serializedAt": [
      {
        "@value": "2014-06-13T23:29:49.729746^^xsd:dateTime"
      }
    ],
    "http://www.w3.org/ns/oa#serializedBy": [
      {
        "@id": "http://localhost/agentID"
      }
    ]
  },
  {
    "@id": "http://localhost/agentID",
    "@type": [
      "http://www.w3.org/ns/prov#SoftwareAgent"
    ],
    "http://xmlns.com/foaf/0.1/name": [
      {
        "@value": "CURL"
      }
    ]
  },
  {
    "@id": "http://example.edu/target/01",
    "@type": [
      "http://purl.org/dc/dcmitype/Text"
    ],
    "http://purl.org/dc/elements/1.1/format": [
      {
        "@value": "html/plain"
      }
    ]
  },
  {
    "@id": "http://localhost/bodyID",
    "@type": [
      "http://www.w3.org/2011/content#ContentAsText",
      "http://purl.org/dc/dcmitype/Text"
    ],
    "http://purl.org/dc/elements/1.1/format": [
      {
        "@value": "text/plain"
      }
    ],
    "http://www.w3.org/2011/content#chars": [
      {
        "@value": "hello there!"
      }
    ]
  }
]
}
```

Figure 9: A JSON-LD Representation of a Text Annotation

```
xml-rdf
<rdf:RDF
  xmlns:ns1="http://www.w3.org/ns/oa#"
  xmlns:ns2="http://purl.org/dc/elements/1.1/"
  xmlns:ns3="http://www.w3.org/2011/content#"
  xmlns:ns4="http://xmlns.com/foaf/0.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <rdf:Description rdf:about="http://localhost/annoID">
    <rdf:type rdf:resource="http://www.w3.org/ns/oa#Annotation"/>
    <ns1:hasTarget rdf:resource="http://example.edu/target/01"/>
    <ns1:hasBody rdf:resource="http://localhost/bodyID"/>
    <ns1:motivatedBy rdf:resource="http://www.w3.org/ns/oa#:commenting"/>
    <ns1:serializedAt>2014-06-13T23:29:49.729746^^xsd:dateTime</ns1:serializedAt>
    <ns1:serializedBy rdf:resource="http://localhost/agentID"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://localhost/agentID">
    <rdf:type rdf:resource="http://www.w3.org/ns/prov#SoftwareAgent"/>
    <ns4:name>CURL</ns4:name>
  </rdf:Description>

   <rdf:Description rdf:about="http://example.edu/target/01">
     <rdf:type rdf:resource="http://purl.org/dc/dcmitype/Text"/>
     <ns2:format>html/text</ns2:format>
   </rdf:Description>

   <rdf:Description rdf:about="http://localhost/bodyID">
    <rdf:type rdf:resource="http://purl.org/dc/dcmitype/Text"/>
    <rdf:type rdf:resource="http://www.w3.org/2011/content#ContentAsText"/>
    <ns2:format>text/plain</ns2:format>
    <ns3:chars>hello there!</ns3:chars>
  </rdf:Description>

</rdf:RDF>
```

Figure 10: A XML-RDF Representation of a Text Annotation

3.7 Tagging A Dataset

Classification of annotations is an important capability primarily to enable users to later discovery information under related thematic areas. OA provides a means to do so via *tagging*. In this case, an annotation is set with the `oa:tagging` motivation. The body of the annotation contains the actual tag. A tag may be a simple text-based keyword or a semantic tag, a URI relating to a concept from a vocabulary. Using the OA semantic tags feature allows annotations to be classified with concepts from SKOS collections. Using the NERC Vocabulary Server[2] it is possible to map classification terms from one vocabulary to similar concepts in other vocabularies.

When inserting a semantic tag the client MUST use the URI of the term as the URI of the body. The body MUST have a type of `oa:SemanticTag` and SHOULD have a `skos:prefLabel`. The label is for the benefit of faceted searches.

2 http://vocab.nerc.ac.uk/

```
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix dctypes: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<chnode:annoID> a oa:Annotation ;
        oa:hasTarget <http://example.edu/dataset/01> ;
        oa:hasBody <http://example.edu/term/01> ;
        oa:motivatedBy oa:tagging ;
        oa:serializedAt "2014-06-13T23:29:49.729746"^^xsd:dateTime ;
        oa:serializedBy <chnode:agentID> .

<chnode:agentID> a prov:SoftwareAgent ;
        foaf:name "CURL" .

<http://example.edu/dataset/01> a dctypes:Dataset .

<http://example.edu/term/01> a oa:SemanticTag ;
        skos:prefLabel "Atmospheric conditions" .
```

Figure 11: A Semantic Tag Annotation

3.8 Linking Datasets

It can often be the case that datasets are somehow related to each other. OA provides a means to associate objects via *linking*. In this case, an annotation is set with the `oa:linking` motivation. Both the body and target are URIs of the datasets.

```
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix dctypes: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<chnode:annoID> a oa:Annotation ;
        oa:hasTarget <http://example.edu/dataset01> ;
        oa:hasBody <http://example.edu/dataset02> ;
        oa:motivatedBy oa:linking ;
        oa:serializedAt "2014-06-13T23:29:49.729746"^^xsd:dateTime ;
        oa:serializedBy <chnode:agentID> .

<chnode:agentID> a prov:SoftwareAgent ;
        foaf:name "CURL" .

<http://example.edu/dataset01> a dctypes:Dataset .

<http://example.edu/dataset02> a dctypes:Dataset .
```

Figure 12: Linked Datasets Annotation

3.9 Citation

This differs from the other examples in the fact that it makes use of the CiTO ontology to create a `CitationAct` and then uses Open Annotation to comment on the citation. The `CitationAct` links the cited and citing entities and the annotation comments on the `CitationAct`.

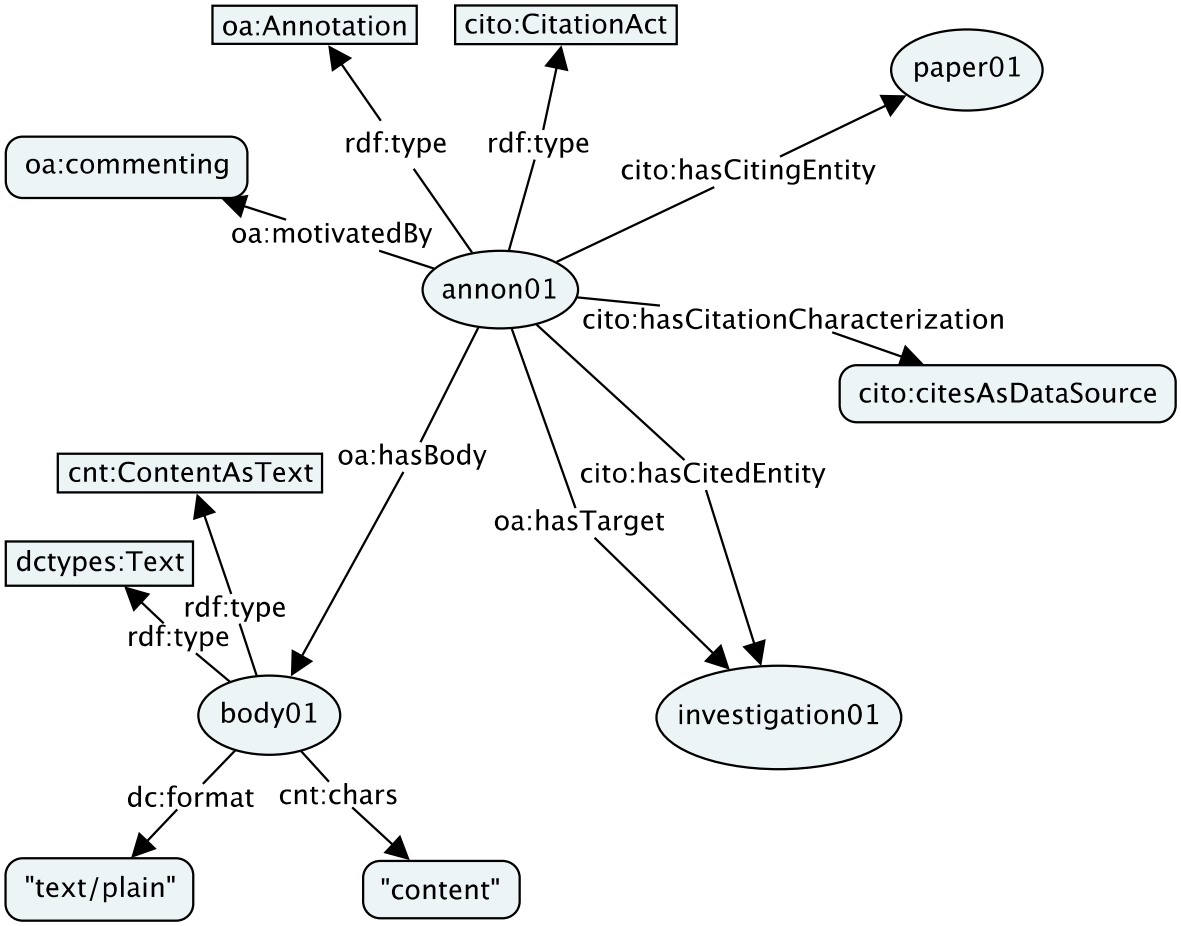

Figure 13: Citing a Dataset and adding a Comment

```
@prefix chnode: <http://localhost/> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix cnt: <http://www.w3.org/2011/content#> .
@prefix cito: <http://purl.org/spar/cito/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix dctypes: <http://purl.org/dc/dcmitype/> .
@prefix fabio: <http://purl.org/spar/fabio/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<chnode:annoID> a oa:Annotation, cito:CitationAct ;
        oa:hasTarget <http://example.edu/dataset/01> ;
        oa:hasBody <chnode:bodyID> ;
        oa:motivatedBy oa:commenting ;
        oa:serializedAt "2014-06-13T23:29:49.729746"^^xsd:dateTime ;
        oa:serializedBy <chnode:agentID> ;
        cito:hasCitingEntity <doi://example.edu/paper/01> ;
        cito:hasCitedEntity <http://example.edu/dataset/01> ;
        cito:hasCitationCharacterization cito:citesAsDataSource .

<chnode:agentID> a prov:SoftwareAgent ;
        foaf:name "CURL" .

<http://example.edu/dataset/01> a dctypes:Dataset .

<chnode:bodyID> a cnt:ContentAsText, dctypes:Text ;
        cnt:chars "This dataset was cited because ..." ;
        dc:format "text/plain" .

<doi://example.edu/paper/01> a fabio:JournalArticle ;
        dcterms:title "My Journal Article" .
```

Figure 14: An Annotation and Citation

4 Calls in More Detail

4.1 Advance_status

The advance_status call can be used to change the status of an annotation. Possible states are 'submitted', 'stable', 'invalid' and 'retired'. The states of 'invalid' and 'retired' are considered to be final states, once an annotation has one of these states then its state can no longer be changed.

If the requested is the same as the existing status it is not an error and has no effect, i.e. if an annotation has a status of 'submitted' and you try and update the status to 'submitted' then there will be no change to the underlying data and you will get a return code of 200.

A user can only update an annotation if they are associated with the annotation or they are a member of the 'moderator' group. Where being associated with an annotation means that one of the `oa:annotatedBy foaf:Person` objects attached to the annotation contains their username.

When an annotation is updated the annotation, and all of the triples with the annotation id as their subject, are moved to the graph for the selected state. The `foaf:Person` object(s) linked via `oa:annotatedBy` are unique so they are also moved. The `foaf:Organization` and `prov:SoftwareAgent` objects are not necessarily unique to that annotation so they are copied. New provenance data about the person requesting the status change is also written out and the `oa:annotatedAt` time is updated.

The call should be a POST and contain a json file, see example file below.

```
{
"annotation": "https://charme-dev.cems.rl.ac.uk/resource/12bfcce3bfe4466e823eb87344ca88b7",
"toState": "retired"
}
```

Figure 15: Json File Used for the Call advance_status

It is possible to use curl to post the file, see example below.

```
curl -X POST 'https://charme-dev.cems.rl.ac.uk:8027/advance_status/' -d@advanceStatus.json
-H 'Authorization: Token 12345678901234567890123456789' -H 'Content-Type:
application/ld+json'
```

Figure 16: Curl Used to Call advance_status

Possible HTTP return codes include:

200 – OK, the annotation was updated, or if the requested status was the same as the existing status nothing was changed

400 – User error, i.e. missing parameter from the json file or trying to change the status of an annotation that is already in a final state

403 – You do not have permission to change the status for this annotation

404 – No annotation was found for the given id

405 – Method not allowed, you did not use a POST request

500 – Oops, something has gone wrong with the server

5 Open Search

5.1 Description Document

Currently the templates in description document are not correct. The correct templates are:

```
search/[atom]?
count={count?}&startPage={startPage?}&startIndex={startIndex?}&bodyType={a0:t
ype?}&citingType={a0:type?}&comment={a1:chars?}&dataType={a0:type?}&domainOfI
nterest={a2:domainOfInterest?}&motivation={a2:motivation?}&organization={a3:n
ame?}&status={a2:status?}&target={a2:target?}&title={a4:title?}&userName={a3:
accountName?}&depth={a2:depth?}&format={a2:format?}
```

```
suggest/[atom]?
count={count?}&startPage={startPage?}&startIndex={startIndex?}&q={searchTerms
?}&bodyType={a0:type?}&citingType={a0:type?}&comment={a1:chars?}&dataType={a0
:type?}&domainOfInterest={a2:domainOfInterest?}&motivation={a2:motivation?}&o
rganization={a3:name?}&status={a2:status?}&target={a2:target?}&title={a4:titl
e?}&userName={a3:accountName?}&depth={a2:depth?}&format={a2:format?}
```

5.2 Search

The `search` will search the graph specified by the value of `status`, if no value is specified for `status` then the 'stable' graph will be used. The search results are filtered based on the values of the parameters:

- `bodyType`
- `citingType`
- `dataType`
- `domainOfInterest`
- `motivation`
- `organization`
- `target`
- `title`
- `userName`

If more than one parameter type is provided, i.e. `dataType=1&domainOfInterest=2,`then the results must match both parameters.

5.2.1 Examples

```
search/atom?count=20&status=submitted&depth=0&format=turtle
```

```
search/atom?count=5&dataType=http://purl.org/dc/dcmitype/Dataset&
domainOfInterest=http://vocab.ndg.nerc.ac.uk/term/P220/1/26&status=submitted
```

5.3 Suggest

The `suggest` will return all of the values for the given search terms from the graph specified by the value of `status`. The search terms are specified via the `q` parameter, allowed values are:

- `bodyType`

- citingType
- dataType
- domainOfInterest
- motivation
- organization
- *

The value for q can be a space separated list e.g. q={dataType motivation}.

The results include the URI and preferred label for each term. The results can be restricted by using values for the parameters listed in 4.2.

N.B. The totalResults refers to the number of different facets found. Currently the number of resources per facet is not reported.

5.3.1 Examples

```
suggest/atom?q={dataType motivation}&status=submitted
```

This will return a list of all of the data types and motivations that are in the submitted graph.

```
suggest/atom?
q={domainOfInterest}&dataType=http://purl.org/dc/dcmitype/Dataset&status=submitted
```

This will return a list of all of the domains of interest that are in the submitted graph where the annotations have a data type of http://purl.org/dc/dcmitype/Dataset.

6 CHARMe Vocabulary

The CHARMe vocablary is available via the `/sparql` endpoint on the node and is contained in the graph `http://localhost:3333/privateds/data/vocab`. It is also available via the vocab endpoint.

7 Security

The Node security provides two components: the node administration and the client OAuth authentication.

The OAuth administration is responsible to grant access, to external client, to protected resources. The OAuth manages four tables accessible through the django admin

Illustration 7.1:

To allow an already registered user to access a specific resource the Node admin insert a new record in the *Clients* table defining:

- the user authorized to the resource
- the RequiredURL the user want to access
- the URL where the user will be redirected if granted
- the Client ID used to identify the user:RequiredURL pair.

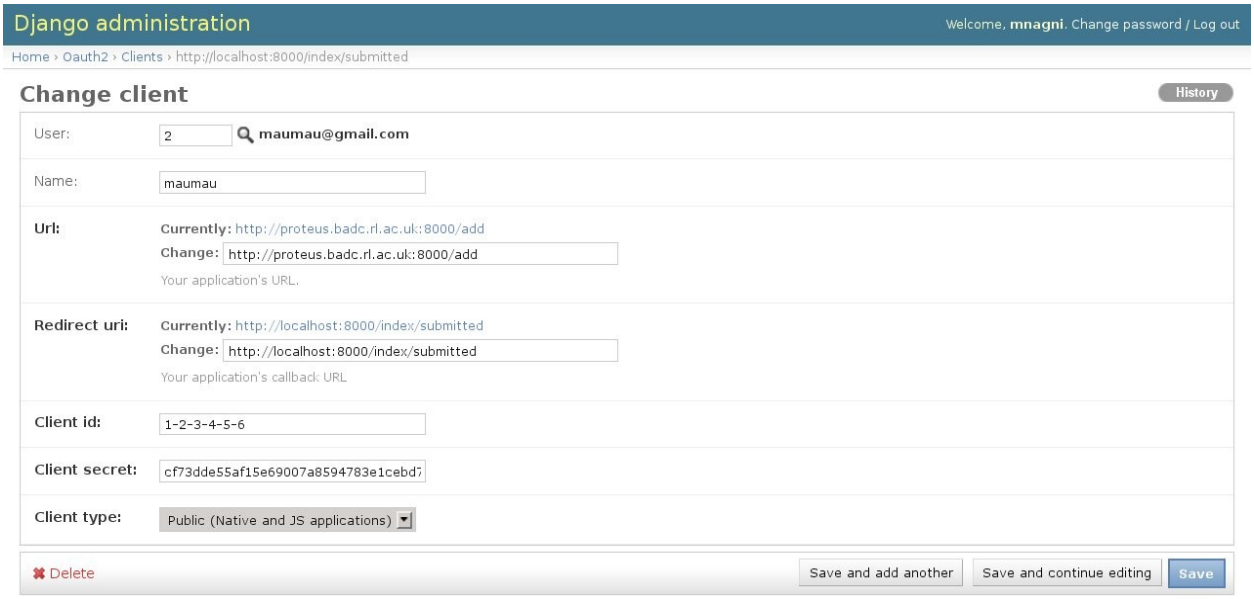

Illustration 7.2:

In a typical request request a client requires an HTTP GET with the following parameters:

?client_id=1-2-3-4-5-6&response_type=token&redirect_uri=http:\\localhost:8000\index\submitted

if the user is not authenticated the node requires user/password. Once authenticated the node returns an HTTP 302 conform to OAuth returning a *access token*, *expiration date*, *type* and *scope* similar to

http://localhost:8000/index/submitted?
access_token=58412ba8bf8d225ed8617fda0806f41a3d1428d0&token_type=Bearer&expires_in=2591999&scope=add_annotation

Unfortunately it doesn't work as expected with ajax calls. For calls via XMLHttpRequest (XHR) transparent redirection works only for urls in the same domain ("same origin"). But proxy software returns 302-responses with urls in "Location" header from other domain ("cross-origin"). So when client code sends a GET-request via XHR (an ajax call) and the user agent gets a 302 response (from proxy) XHR fails the request. The result contains absolutely nothing that could help to identify the feailure reason. In a jQuery's "fail" handler we'll get status: 0, responseText:"",statusText:"error".

That's very bad. A conclusion that we can make is that Single Page Applications can be incompatible with reverse proxy software.

One point in the HTTP spec can help here. It states that: So there should be no transparent redirections for POST requests. Our application on getting an error with status=0 can send a POST-request to its

server. And the proxy software will not return 302 responses as it can (and will) be ignored by the user agent. Instead it'll return a 401 response asking the user agent to authenticate. So the user will be prompted for credentials and if they're ok the proxy will pass the original request and all subsequent ones. Application continues to work.