# Dacura Linked Data API

All API calls go to:   
[http://dacura.install.url/[collectionid]/[ldtype](http://dacura.install.url/%5bcollectionid%5d/%5bldtype)]  
where ldtype = candidate|ontology|graph|schema|ld

4 types of json objects are used:

**Dacura Linked Data Object**

**Dacura Linked Data Object Update**

**Dacura API Result Object** which may contain one or more:

**Dacura Graph Update Result Object**

Contents

[Dacura Linked Data API 1](#_Toc444996758)

[1. List Linked Data Objects 1](#_Toc444996759)

[Arguments 1](#_Toc444996760)

[Return 1](#_Toc444996761)

[2. List Linked Data Object Updates 1](#_Toc444996762)

[Arguments 1](#_Toc444996763)

[Return 1](#_Toc444996764)

[3. View Linked Data Object 1](#_Toc444996765)

[3b. View Linked Data Object Fragment 1](#_Toc444996766)

[4. Create Linked Data Object 1](#_Toc444996767)

[5. Update Linked Data Object 1](#_Toc444996768)

[5b. Update Linked Data Object Fragment 1](#_Toc444996769)

[6. Delete Linked Data Object 1](#_Toc444996770)

[6b Delete Linked Data Object Fragment 1](#_Toc444996771)

[7. View Linked Data Object Update 1](#_Toc444996772)

[8. Update Linked Data Object Update 1](#_Toc444996773)

[9. Delete Linked Data Object Update 1](#_Toc444996774)

## 1. List Linked Data Objects

GET / - *retrieve listing of linked data objects*

Argumentsoptions *array* options to change contents of what comes back

[

include\_all *boolean* set to 1 to include contents with each listed object   
status *string pending|reject|accept* if set, only objects with this status are returned  
version *integer* if set, only objects which currently have this version are returned

]

**Example** [http://dacura.cs.tcd.ie/rest/seshat/ontology?options[include\_all]=1&options[status]=reject](http://dacura.cs.tcd.ie/rest/seshat/ontology?options%5binclude_all%5d=1&options%5bstatus%5d=reject) (list all objects, including their contents, that have been rejected).

Return

**A**n array of objects, each having the following fields: [{

id *string* the object’s id  
collectionid *string* the id of the collection that the object belongs to (*all* for system)  
version *integer* the current version of the object  
type *string ontology|candidate|graph|schema* type of object   
url *string* canonical url of object (dacura/collection\_id/type/id)  
createtime *integer* Unix timestamp of object creation  
modtime *integer* Unix timestamp of last object update  
status *string* *pending|reject|accept* current status of object  
meta *object* meta-data array, contents specific to object type  
size *integer* size in bytes of object contents as stored on disk   
contents *object* contents json object – **only included if include\_all is set in options**

}]

**Example**

[{"id":"adms","collectionid":"all","version":"6","type":"ontology","status":"accept","createtime":"1444839098","modtime":"1453994979","meta":{}}]

### Error

HTTP Error code plus text message in message body

## 2. List Linked Data Object Updates

GET /update *- retrieve listing of updates to linked data objects*

Argumentsoptions *array* options to change contents of what comes back

{

include\_all *boolean* set =1 to include meta, forward & backward with each listed update  
status *string pending|reject|accept* if set, updates with this status only returned  
to\_version *integer* if set, only updates with this to\_version number will be returned  
from\_version *integer* if set, only updates with this from\_version number will be returned  
targetid *string* if set, only updates to the object with this id will be returned

}

**Example:** http://dacura.cs.tcd.ie/rest/seshat/candidate?options[include\_all]=1&options[to\_version]=0& options[targetid]=rome (list all updates, including their contents, on the rome object that have not been accepted).

Return

An array of update objects, each having the following fields:  
[{

eurid *string –* the id of the update object   
targetid *string –* the id of the object being updated  
collectionid *string –* the id of the collection the object belongs to   
from\_version *integer –* the version of the object that the update was applied to  
to\_version *integer –* the version of the object that the update created (0 if NA)  
type *string –* the linked data type of the object (ontology, graph, candidate)  
createtime *integer –* Unix timestamp of update creation  
modtime *integer –* Unix timestamp of last update modification  
status *integer –* status of update (accept, reject, pending  
size *integer* size in bytes of update contents as stored (forward & backward)  
meta *object* – meta-data json of update *only included if include\_all is set in options*  
forward *object* – update command in json format *only if include\_all is set in options*  
backward *object* – undo command in json format *only if include\_all is set in options*

}]

**Example**

[{"eurid":"7","targetid":"prov","type":"ontology","collectionid":"all","status":"accept","from\_version":"4","to\_version":"5","createtime":"1442528528","modtime":"1442528528","size":"142"}]

## 3. View Linked Data Object

GET /[object\_id] *– retrieve representation of a linked data object*

Argumentsversion *integer* theversion of the object desired (default is latest version)

format *string* the format that the contents of the object should be returned in.

Must be one of LDO::$valid\_display\_types:

"json" => "Internal JSON Format",

"html" => "internal html view",

"jsonld" => "JSON LD",

"turtle" => "Turtle Terse RDF",

"rdfxml" => "RDF/XML format",

"n3" => "Notation3",

"triples" => "Triples",

"ntriples" => "N-Triples",

"quads" => "Quads",

"nquads" => "N-Quads",

"dot" => "Graphviz Dot Notation",

"png" => "Portable Network Graphics (PNG)",

"gif" => "Graphics Interchange Format (GIF)",

"svg" => "Scalable Vector Graphics (SVG)",

options *array* optional flags to support dacura management system (not available to outside world)

{

history *boolean* set =1 to include object version history listing with object

updates *boolean* set =1 to include updates listing for object with object

analysis *boolean* set =1 to include object analysis results with object

ns *boolean* set =1 to use namespace prefixes in urls

plain *boolean* set =1 to have output printed without html styling

addressable *boolean* set =1 to use addressable urls for blank node ids

}

**Example:** [http://dacura.cs.tcd.ie/rest/seshat/candidate/rome?version=1&format=turtle&options[history]=1&options[ns]=1](http://dacura.cs.tcd.ie/rest/seshat/candidate/rome?version=1&format=turtle&options%5bhistory%5d=1&options%5bns%5d=1) fetch version 1 of the rome candidate in the turtle format, using namespace-prefixed urls, including history listing.

### Return

A json object, representing the linked data object, with the following fields: {  
 id *string* the id of the object ldtype *string candidate|ontology|graph|schema* linked data type of object  
options *array* echoed options array from inputformat *string* formatof contents (echoed from argument)  
content *mixed* contents themselves (string or json object)  
 meta *object* meta-data array containing the following fields

{

cid *string* the id of the collection the object belongs to  
version *integer* the version number of this version  
latest\_version *integer* the latest version of the object  
created *integer –* Unix timestamp of object creation  
modified *integer –* Unix timestamp of last modification  
version\_created *integer –* Unix timestamp of when this version was created  
status *string – pending|reject|accept* status of this version  
latest\_status *string – pending|reject|accept* status object’s latest version  
compressed *boolean –* set to true if urls have namespace prefixes  
addressable\_bnids *boolean –* set to true if blank nodes have addressable urls

}  
historyarray of previous version of this object (only if options[history] is set)  
analysis *string* html result string of object analysis (only if options[analysis] is set)  
updates *array* of updates to this object (only if options[updates] is set)

}

### Errors

## 3b. View Linked Data Object Fragment

GET /[object\_id]/[fragment\_id] – fetches a fragment of an object

This function works exactly the same as the above with the only difference being in the following return fields{

content *mixed* containsfragment contents, not object contents   
fragment\_id *string* the id of the fragment / blank node  
fragment\_path *array* the path to the fragment in the object (for embedded objects)

}

**Example**

[http://dacura.cs.tcd.ie/rest/seshat/candidate/rome/religion?version=1&format=jsonld&options[ns]=1](http://dacura.cs.tcd.ie/rest/seshat/candidate/rome/religion?version=1&format=jsonld&options%5bns%5d=1) fetch version 1 of the fragment with id “religion” from the rome candidate object in the jsonld format, using namespace-prefixed urls.

{"id":"rome","ldtype":"candidate","meta":{"cid":"seshat",version":"1","created":"1456353029","modified":"1456353029","latest\_status":"pending","latest\_version":"2","version\_created":"1456353029","compressed":false,"addressable\_bnids":true,"status":"pending"},"contents":{"http:\/\/localhost\/dacura\/all\/ontology\/rome\/religion":{"http:\/\/an.example\/predicate1":{"type":"http:\/\/www.w3.org\/2001\/XMLSchema#string","data":"object1"}}},"format":"json","options":{"ns":"1"},"fragment\_id":"religion","fragment\_path":{“rome:”: {“seshat:hasReligiousSystem”: {}}}}

## 4. Create Linked Data Object

POST / - create a new linked data object

### Arguments

JSON object representing the new **Dacura Linked Data Object** is sent as a json encoded object in the post request, with the HTTP header: Content-Type application/json and the following fields (all fields are optional)

{

[$demand\_id\_token] *string -* the id requested for this object

Note: “@id” is the default demand id token, but it can be changed in collection configuration.

options *array -* optional flags to support dacura management system (not available to outside world)

{

show\_result *boolean* set =1 to include created object in result message

show\_dqs\_triples *boolean* set =1 to include deltas to dqs graph in result

show\_ld\_triples *boolean* set =1 to include deltas to object in result

fail\_on\_id\_denied *boolean* if=1 the call will fail if any requested ids are denied

}

format *string* what format are the contents in?

if omitted, system will auto-detect. Must be one of LDO::$valid\_input\_types:

"json" => "Internal JSON Format",

"jsonld" => "JSON LD",

"turtle" => "Turtle Terse RDF",

"rdfxml" => "RDF/XML format",

"n3" => "Notation3",

"triples" => "Triples",

"ntriples" => "N-Triples",

"quads" => "Quads",

"nquads" => "N-Quads"

contents *mixed –* the contents of the object – a json object if format is json type, string otherwise. *Note only one of* {ldurl*,* contents*,* ldfile} *should be present in the request.*

ldurl *string –* A URL where the contents of the object can be found – the format should be whatever is defined in the *format* field. *Note only one of* {ldurl*,* contents*,* ldfile} *should be present in the request.*

ldfile *string –* Path to uploaded file where the contents of the object can be found – the format should be whatever is defined in the *format* field. *Note only one of* {ldurl*,* contents*,* ldfile} *should be present in the request.*

meta *object –* json object containing new object’s metadata settings. All LD objects support the following meta-data fields:

{

image *string* URL of image file that will represent this object

title *string* Title text which will represent object in management screens

url *string* URL of canonical url for external representation of this object – used for ontologies to point to their purl / etc.

status *string* (pending|reject|accept)requested status of new object - subject to policy approval and DQS validation. If status=pending, allows objects to be created and stored without them passing dqs validation. Default is accept.

}

test *boolean –* if set to 1, no state updates will take place, the creation will be run as a test

}

### Return

**A DacuraResult JSON object**, representing the result of the object creation, containing the following properties. The HTTP response code should match the status field of the response object. 200 = accept, 202 = pending, >202 = reject. The body of the response should always contain a DacuraResult object, even when the HTTP response code indicates an error.

{

status *string* pending|reject|accept – what happened to the new object – was it published to the graph (accept), published to the object store (pending) or rejected altogether (reject)

message *mixed (string|object) –* the message tobe reported back to the user. May be structured (with {body, title, extra} fields or a simple string.

test *boolean –* was this a test creation – echoes test setting from input

errors *array* – array of errors to report to user

warnings *array* – array of warnings to report to user

result *object –* Dacura Linked Data Object (as returned by API.3) that was/would be created by the operation. Only present if show\_result is set in input options

graph\_ld *object –* **Dacura Graph Update Result Object –** representing the update to the linked data object as inserted / deleted triples. Only present if show\_ld\_triples is set in input options

graph\_dqs *object –* **Dacura Graph Update Result Object –** representing the update to the DQS graph representation of the object as inserted / deleted triples. Only present if show\_dqs\_triples is set in input options

}

A **Dacura Graph Update Result Object** contains the following properties:

{

hypothetical *boolean –* iftrue the result described in the object did not actually take place, they describe what would happen if the object was published to the graph. If false, any changes described actually took place.

status *string [pending|reject|accept] –* describes the result of the validation decision for this graph (if no validation took place, status is accept).

test *boolean –* was this result in response to a test invocation? If test = true, then hypothetical should also be true, although the reverse is not true – if test is false and the object fails validation at an early stage, the graph result may be hypothetical without it being a test case. This should always echo the overall DacuraResult test setting.

inserts *array* – arrayof triples / quads representing the inserts into the graph(s) caused by the request.

deletes *array* – arrayof triples / quads representing the deletes from the graph(s) caused by the request.

}

## 5. Update Linked Data Object

POST /[object\_id]- update existing linked data object with id *object\_id*

### Arguments

JSON object representing the updated **Dacura Linked Data Object** is sent as a json encoded object in the post request, with the HTTP header: Content-Type application/json and the following fields (all fields are optional)

{

options *array -* optional flags to support dacura management system (not available to outside world)

{

show\_result *int* set =1 to include updated LDO in result field,   
set=2 to include LDOUpdate in result field

show\_dqs\_triples *boolean* set =1 to include deltas to dqs graph in result

show\_ld\_triples *boolean* set =1 to include deltas to object in result

fail\_on\_id\_denied *boolean* if=1 the call will fail if any requested ids are denied

show\_meta\_delta *boolean* set =1 to include deltas to object meta-data

show\_update\_triples *boolean* set =1 to include deltas to deferred updates

}

format *string* what format are the submitted contents in?

if omitted, system will auto-detect. Must be one of LDO::$valid\_input\_types:

"json" => "Internal JSON Format",

"jsonld" => "JSON LD",

"turtle" => "Turtle Terse RDF",

"rdfxml" => "RDF/XML format",

"n3" => "Notation3",

"triples" => "Triples",

"ntriples" => "N-Triples",

"quads" => "Quads",

"nquads" => "N-Quads"

contents *mixed –* the contents of the object – a json object if format is json type, string otherwise. *Note only one of* {ldurl*,* contents*,* ldfile} *should be present in the request.*

ldurl *string –* A URL where the contents of the object can be found – the format should be whatever is defined in the *format* field. *Note only one of* {ldurl*,* contents*,* ldfile} *should be present in the request.*

ldfile *string –* Path to uploaded file where the contents of the object can be found – the format should be whatever is defined in the *format* field. *Note only one of* {ldurl*,* contents*,* ldfile} *should be present in the request.*

meta *object –* json object containing updates to object’s metadata settings.

umeta *object –* json object containing meta-data about the update itself – currently just status is supported

test *boolean –* if set to 1, no state updates will take place, the creation will be run as a test

editmode *string* either *update* or *replace* – defines how the incoming object should be treated, either as a replacement for the stored object or as an update instruction

}

### Return

**A DacuraResult JSON object**, representing the result of the object update, containing the following properties. The HTTP response code should match the status field of the response object. 200 = accept, 202 = pending, >202 = reject. The body of the response should always contain a DacuraResult object, even when the HTTP response code indicates an error.

{

status *string* pending|reject|accept – what happened to the update – was it published in the object store (and maybe the graph too) (accept), published to the deferred update queue (pending) or rejected altogether (reject)

message *mixed (string|object) –* the message tobe reported back to the user. May be structured (with {body, title, extra} fields) or a simple string.

test *boolean –* was this a test update – echoes test setting from input

errors *array* – array of RVO errors to report to user

warnings *array* – array of RVO warnings to report to user

result *object | string –* **Dacura Linked Data Object** (as returned by API.3) that was/would be updated by the operation (show\_result=1) OR a **Dacura Linked Data Update Object** (as returned by API.7) (show\_result = 2) Only present if show\_result is set in input options

graph\_ld *object –* **Dacura Graph Update Result Object –** representing the update to the linked data object as inserted / deleted triples. Only present if show\_ld\_triples is set in input options

graph\_update *object –* **Dacura Graph Update Result Object –** representing the update to the deferred object queue as inserted / deleted triples. Only present if show\_ld\_triples is set in input options

graph\_dqs *object –* **Dacura Graph Update Result Object –** representing the update to the DQS graph representation of the object as inserted / deleted triples. Only present if show\_dqs\_triples is set in input options

graph\_meta *object* - **Dacura Graph Update Result Object-** representing the update to the object’s meta data. Only present if show\_meta\_data is set in input options

}

## 5b. Update Linked Data Object Fragment

POST /[object\_id]/[fragment\_id]– updates a fragment of an object

This function works exactly the same as the above with the only difference being in the following return fields of the changed and original ldos{

content *mixed* containsfragment contents, not object contents   
fragment\_id *string* the id of the fragment / blank node  
fragment\_path *array* the path to the fragment in the object (for embedded objects)

}

The result itself will have the fragment\_id field set to indicate it is a fragment update

**Example**

6. Delete Linked Data Object

DELETE /[object\_id]

## 6b Delete Linked Data Object Fragment

DELETE /[object\_id]/[fragment\_id]

## 7. View Linked Data Object Update

GET /update/[update\_id]

Argumentsversion *integer* theversion of the object that the update should be applied to – if not  
 present, the version that the update was made against is used – set to 0 to apply  
 update to latest version

format *string* the format that the contents of the update should be returned in.

Must be one of LDO::$valid\_display\_types:

"json" => "Internal JSON Format",

"html" => "internal html view",

"jsonld" => "JSON LD",

"turtle" => "Turtle Terse RDF",

"rdfxml" => "RDF/XML format",

"n3" => "Notation3",

"triples" => "Triples",

"ntriples" => "N-Triples",

"quads" => "Quads",

"nquads" => "N-Quads",

"dot" => "Graphviz Dot Notation",

"png" => "Portable Network Graphics (PNG)",

"gif" => "Graphics Interchange Format (GIF)",

"svg" => "Scalable Vector Graphics (SVG)",

options *array* optional flags to support dacura management system (not available to outside world)

{

original *boolean* set =1 to include original version of object before update applied

changed *boolean* set =1 to include changed version of object with update applied

analysis *boolean* set =1 to include update analysis results with update

ns *boolean* set =1 to use namespace prefixes in urls

pretty *boolean* set =1 to have output pretty printed

addressable *boolean* set =1 to use addressable urls for blank node ids

}

**Example:** [http://dacura.cs.tcd.ie/rest/seshat/candidate/rome/update/345?version=1&format=turtle&options[changed]=1&options[ns]=1](http://dacura.cs.tcd.ie/rest/seshat/candidate/rome/update/345?version=1&format=turtle&options%5bchanged%5d=1&options%5bns%5d=1) fetch update 345 to the rome candidate, apply it against version 1 of the candidate and include the changed version in the result, in the turtle format, using namespace-prefixed urls.

### Return

A json object, representing the linked data update object, with the following fields: {  
 id *string* the id of the update  
 ldtype string the ldtype of the object being updated  
target\_id string the id of the object being updated  
options *array* echoed options array from inputformat *string* formatof contents (echoed from argument)  
insert *mixed* parts added to object (format governs format)  
 delete *mixed* parts deleted from object (string or json object)  
 meta *object* meta-data array containing the following fields

{

cid *string* the id of the collection the object belongs to  
from\_version *integer* the version that the update was made against  
to\_version *integer* the version that the update was made against  
created *integer –* Unix timestamp of update creation  
modified *integer –* Unix timestamp of last modification  
status *string – pending|reject|accept* status of the update  
compressed *boolean –* set to true if urls have namespace prefixes  
addressable\_bnids *boolean –* set to true if blank nodes have addressable urls

}  
historyarray of previous version of this object (only if options[history] is set)  
analysis *string* html result string of object analysis (only if options[analysis] is set)  
updates *array* of updates to this object (only if options[updates] is set)

}

### Errors

$this->getPropertiesAsArray()

## 8. Update Linked Data Object Update

## 9. Delete Linked Data Object Update