
ADL-R Learning Registry

POST Interface Specification to the ADL Registry Version 1.6



Developed For:

Advanced Distributed Learning Initiative
and
Defense Technical Information Center (DTIC)
8725 John J. Kingman Highway
Fort Belvoir, Virginia 22060-6218

Revised By:

ADL-R User Support Team
support@adlregistry.org
<https://adlregistry.dtic.mil>
1.888.DOD.ADLR

Knowledge Media, Inc.
3 Bethesda Metro Center
Suite 700
Bethesda, MD 20814

Revised August 1, 2006

I) POST Interface Definition

The POST submission to the registry has to be a multipart/form-data encoded post.

GET transactions are not supported.

The POST has to contain a combination of input fields defined below:

command: the command that the ADL Registry transaction processor is to perform. There are currently four commands that can be issued:

- processtransaction
- gettransactionstatus
- gettransactionid
- searchrequest

userid: a data input field to specify the user's handle identifier.

password: a data input field to specify the user's password.

email: a data input field to specify the email address where the ADL Registry transaction processor is to send the submission's status log once it has completed processing it.

transactionid: a data input field to specify the transaction identifier that a user wants the ADL Registry transaction process to use to identify its submission requests.

format: a data input field to specify the return format of the ADL Registry transaction processor's status. There are currently only two options: application/xml or application/html.

registry: a data input field to specify which registry the ADL Registry transaction processor is to interact with. This value shall be a handle.

batchfile: a file input field to contain the contents of the batch submission file to upload to the ADL Registry transaction processor.

textsearch: a data input field to contain a search expression for the registry's to evaluate. This query field should contain a Lucene compatible search query.

The cardinality of the field is defined according to the type of command issued and is as follows:

command	userid	password	email	transactionid	format	registry	textsearch	batchfile
processtransaction	1	1	1/0	1/0	1/0	1	n/a	1
gettransactionstatus	1	1	n/a	1	1/0	1	n/a	n/a
gettransactionid	1	1	n/a	n/a	1/0	1	n/a	n/a
searchrequest	n/a	n/a	n/a	n/a	1/0	1	1	n/a

This section describes each of the four requests that a client can issue against the ADL Registry Transaction Process (ARTP).

1) processtransaction

This command is used to submit a batch submission to the registry's transaction processor formatted according to the adlreg-transaction.xsd schema. The ARTP will first authenticate the client. If the client is properly authenticated, the ARTP will then verify and validate the XML submission. If the XML is valid, the ARTP will then return the user an acknowledgment of a successful submission and a transaction id and start processing the batch submission.

The user can at anytime request a process status log by querying the registry using the gettransactionstatus command described in a later section of this document.

When the transaction processor completes processing the batch, it will send the status log to the user specified email address (if one has been supplied).

The ADL-R requires that a processtransaction xml submission be processed through a virus checker interim POST interface. The POST information remains the same, but the transaction will be made to a different URL which is:

<https://adlregistry.dtic.mil/postinterface/processtransaction.cfm>

The fields expected by the ARTP when receiving a *processtransaction* command are as follows:

- command: processtransaction
- userid: the user is required to specify a valid, authenticated and authorized user identifier. This identifier has to be a handle.
- password: the user is required to specify a password to enable the transaction process to authenticate the user.
- registry: the user is required to specify the registry that it wants to target to store and index the metadata records contained in the transaction request.
- email: the user can optionally specify an email address where the ARTP is to send a notification of the completion of the batch submission's processing.
- format: the user can optionally specify the format of the returns to xml or html by setting the field's value to application/xml or application/html. If this field is not specified, it will return the results in XML as an instance of the ADL-R-transaction-status.xsd
- batchfile: the user is required to provide a file input to upload the batch submission to the ARTP.
- transactionid: the user can optionally specify a transaction id for the ARTP to associate with the submitted batch. This transaction id cannot be generated by the client and has to have been exclusively acquired using the gettransactonid command. A specific transaction id cannot be reused and would result in the processtransaction request to fail.

2) gettransactionstatus

This command enable the user to request the status of its batch submission at any point after it submitted it for processing. The user has to authenticate itself, target a specific

registry and provide the transaction id for the batch it wants to receive a status log for. The status log returned will be an instance of the ADL-R-transaction-status.xsd.

- command: gettransactionstatus
- userid: the user is required to specify a valid, authenticated and authorized user identifier. This identifier has to be a handle and is required.
- password: the user is required to specify a password to enable the transaction process to authenticate the user. This field is required.
- registry: the user is required to specify the registry that it wants to target with a query for the status of its submission. This should be the same as that used in the *processtransaction* request.
- format: the user can optionally specify the format of the transaction status returns to xml or html by setting the field's value to application/xml or application/html. If this field is not specified, it will return the results in XML as an instance of the ADL-R-transaction-status.xsd
- transactionid: the user is required to specify the id it wants to get the transaction status of.

3) gettransactionid

This command enables a user to request a transaction id before submitting a *processtransaction* request.

- command: gettransactionid
- userid: the user is required to specify a valid, authenticated and authorized user identifier. This identifier has to be a handle.
- password: the user is required to specify a password to enable the transaction process to authenticate the user.
- registry: the user is required to specify the registry that it wants to acquired a transaction id from. This should be the same as that which will be used in the *processtransaction* request.
- format: the user can optionally specify the format of the transaction status returns to xml or html by setting the field's value to application/xml or application/html. If this field is not specified, it will return the results in XML as an instance of the ADL-R-transaction-status.xsd. The current version 1.6 of the registry only supports XML.

4) searchrequest

This command enables a user to request a search against a specific metadata registry.

- command: searchrequest
- textsearch: this field enables the user to specify a search against the ADL registry. The user can specify a complex search by using the Lucene search syntax.
- registry: the user is required to specify the registry that it wants to issue a search against.
- format: the user can optionally specify the format of the search results to xml or html by setting the field's value to application/xml or application/html. If this field is not specified, it will return the results in html.

The ADL Registry Lucene Search interface supports a number of indexed and dynamic search keys. The following is a list of indexed fields available for use:

title
description
keyword
version
status - (/lom/lifeCycle/status/source)
statusValue - (/lom/lifeCycle/status/value)
role - (/lom/lifeCycle/contribute/role/source)
metadataSchema
format
rights - (/lom/rights/copyrightAndOtherRestrictions/source)
rightsValue - (/lom/rights/copyrightAndOtherRestrictions/value)
purpose - (/lom/classification/purpose/value)
taxon - (/lom/classification/taxonPath/source/string)

A typical sample query is:

title: ADL

In addition to indexed fields, dynamic keys have been created that support search with specific metadata elements:

/lom/lifeCycle/contribute/role/value/author
/lom/lifeCycle/contribute/role/value/publisher
/lom/lifeCycle/contribute/role/value/unknown
/lom/lifeCycle/contribute/role/value/initiator
/lom/lifeCycle/contribute/role/value/terminator
/lom/lifeCycle/contribute/role/value/validator
/lom/lifeCycle/contribute/role/value/editor
/lom/lifeCycle/contribute/role/value/graphical_designer
/lom/lifeCycle/contribute/role/value/technical_implementer
/lom/lifeCycle/contribute/role/value/content_provider
/lom/lifeCycle/contribute/role/value/technical_validator
/lom/lifeCycle/contribute/role/value/educational_validator
/lom/lifeCycle/contribute/role/value/script_writer
/lom/lifeCycle/contribute/role/value/instructional_designer
/lom/lifeCycle/contribute/role/value/subject_matter_expert
/lom/classification/purpose/value/security_level
/lom/classification/purpose/value/content_type
/lom/classification/purpose/value/distribution_restrictions
/lom/classification/purpose/value/conforms_to
/lom/classification/purpose/value/discipline
/lom/classification/purpose/value/idea
/lom/classification/purpose/value/prerequisite
/lom/classification/purpose/value/educational_objective
/lom/classification/purpose/value/accessibility_restrictions
/lom/classification/purpose/value/educational_level

/lom/classification/purpose/value/skill_level/lom/classification/
purpose/value/competency

A typical sample query using a dynamic key is:

/lom/lifeCycle/contribute/role/value/author: ADL

To determine the specific schema elements and vocabulary, please refer to the ADL-R Cardinality documents and examples in the ADL-R Portal Library at:

<https://adlregistry.dtic.mil/5/index1.htm>

III} Creating a Custom Query Interface

A custom POST query interface can be created using the interface definitions in this document. Since the results may be returned in XML, they can easily be parsed and processed to perform local operations or data imports.

Examples of custom POST interfaces to ADL-R include:

- LCMS REG-T automated ProcessTransaction Inserts
- Scheduled search queries to support content alerts or content update summaries
- Scheduled GetTransactionStatus to verify current workflow status

Sample POST form examples are available below:

Here is a Practice Registry example with correct POST parameters:

[http://www.knowledge-
media.com/synergy/contentexport/651/search/search_simplepostexample_practiceregistry.htm](http://www.knowledge-media.com/synergy/contentexport/651/search/search_simplepostexample_practiceregistry.htm)

Here is a Operational Registry Example with correct POST parameters:

[http://www.knowledge-
media.com/synergy/contentexport/651/search/search_simplepostexample_operationalregistry.htm](http://www.knowledge-media.com/synergy/contentexport/651/search/search_simplepostexample_operationalregistry.htm)

SAMPLE POST SEARCH REQUEST FORM FOR PRACTICE REGISTRY

```
<html>
<head>
<title>Search Content Example- Practice Registry</title>

</head>

<body>

<div align="center">
<form enctype="multipart/form-data" method="post"
action="http://64.14.115.157/CORDRAWeb/processrequest" name=deposit>
  <input type=hidden name="command" value="searchrequest">
  <tbody><tr>
    <td width="188" align="right" valign="baseline" height="23"><label
for="TextSearch"><strong>Search Content</strong></label>.</td>
    <td width="398" colspan="1" align="center" valign="baseline" height="23">
      <input type="input" name="textsearch" size="45" id="TextSearch">&nbsp;  <input
type="submit" value="Search">
    </td>
    <td width="150" valign="top" height="13">
      <div align="center">
        <center>

        </center>
      </div>
    </td>
  </tr>
  <input type="hidden" name="registry" value="4444/registry">
</tbody>
</form>

</div>

</body>
</html>
```

SAMPLE POST SEARCH REQUEST FORM FOR OPERATIONAL REGISTRY

```
<html>
<head>
<title>Search Content Example- Operational Registry</title>

</head>

<body>

<div align="center">
<form enctype="multipart/form-data" method="post"
action="http://132.151.9.94:7001/CORDRAWeb/processrequest" name=deposit>
    <input type=hidden name="command" value="searchrequest">
    <tbody><tr>
        <td width="188" align="right" valign="baseline" height="23"><label
for="TextSearch"><strong>Search Content</strong></label>:</td>
        <td width="398" colspan="1" align="center" valign="baseline" height="23">
            <input type="input" name="textsearch" size="45" id="TextSearch">&nbsp;<input
type="submit" value="Search">
        </td>
        <td width="150" valign="top" height="13">
            <div align="center">
                <center>

                </center>
            </div>
        </td>
    </tr>
    <input type="hidden" name="registry" value="100.3/registry">
    </tbody>
</form>

</div>

</body>
</html>
```