Learning Registry
Quick Reference Guide

Document Version: 1.0
Last Edited: 17 October, 2011

PENDING UPDATE, USE THIS INFORMATION WITH CARE

Prepared by:
John Brecht, SRI International (john.brecht@sri.com)

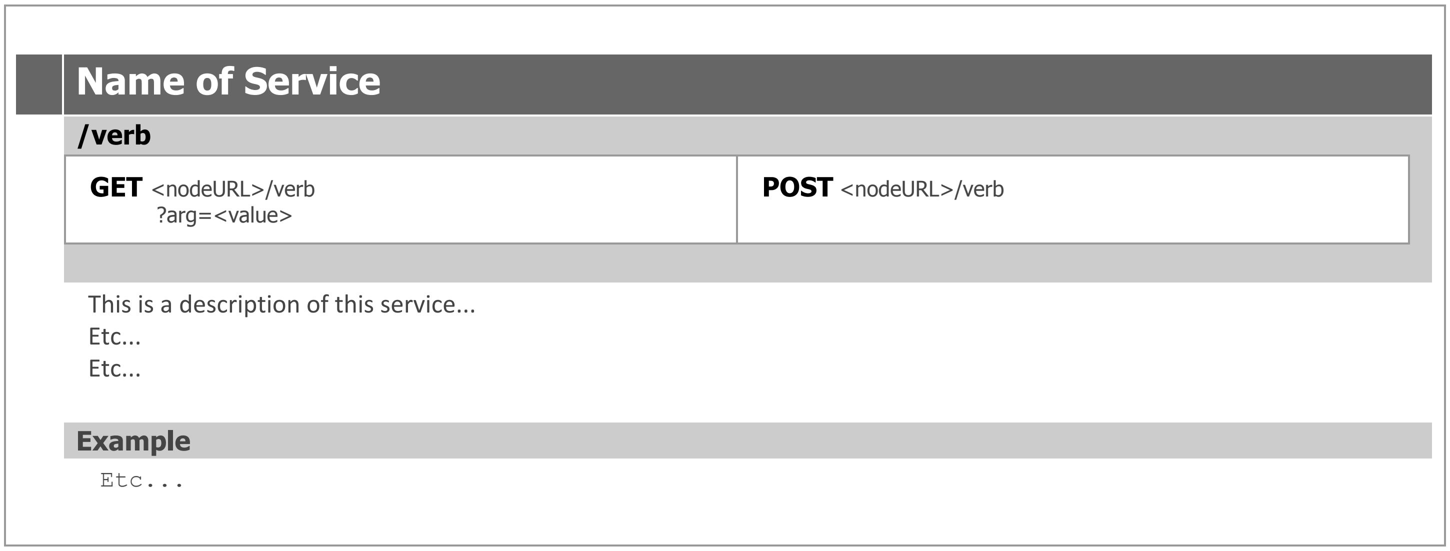
Learning Registry
Quick Reference Guide

Contents

Introduction

Welcome to the Learning Registry! The purpose of this document is to provide a brief reference to the principal data structures and services that typical users of the Learning Registry will most frequently interact with. Administrative users (users standing up or maintaining Learning Registry nodes) and Learning Registry core developers should consult the Learning Registry Technical Specification for complete documentation of all Learning Registry features both internal and external. This document is intended to be consistent with the *Learning Registry Technical Specification*.

The services are list by purpose (publish, retrieve, etc.) Each has a description, samples and additional information. *Figure 1* is an example of a section describing a service. The start of each section contains the syntax for HTTP GET, and/or HTTP POST, as appropriate. Example code is illustrated with cURL command lines. Code examples have additional formatting for the purposes of readability.



Name of Service

/verb

GET <nodeURL>/verb ?arg=<value>	**POST** <nodeURL>/verb

This is a description of this service...
Etc...
Etc...

Example

```
Etc...
```

Figure 1. Example description of a service topic.

Data Model

Conceptual Design

The Learning Registry data model describes resources, metadata, paradata, etc., that are distributed by the network. Data read from, or written to, the Learning Registry by users is referred to as a resource data description. The following definitions are necessary before proceeding:

Resource: Some document, media, web site, etc., that is designed for, or has been used, in an educational context. Within the Learning Registry it typically refers to digital resources. In principle, physical resources such as places, artifacts, meeting rooms, equipment and so forth may also be documented.

Resource Data: Data that describes a learning resource. Types of resource data include, but is not limited to, metadata and paradata.

> **Resource Data Description**: A document submitted to, or retrieve from, the Learning Registry that serves as notification to the existence of a resource, metadata or paradata document within the network. It provides some insight as to the properties of the document it refers to and the resource that document describes. Data read from, or written to, the Learning Registry by users is a *resource data description*.

Its important to emphasize that the Learning Registry does not (typically) contain learning resources. Nor does it contain metadata or paradata about learning resources. Rather, it contains notifications indicating the *existence* of resources, metadata or paradata.

Most users will be familiar with metadata, in this context, as describing the *properties* of a learning resource. Examples of such could be the author, subject, level, etc. Paradata, on the other hand, describe the contextualized *use* of a learning resource. Examples of paradata could be: the number of times used by some community, ratings, comments, etc.

The Learning Registry is agnostic as to the format of metadata or paradata referenced by a resource data description. The referenced document can be delivered inline, linked, or attached to a resource data description:

inline: Resource data is in an object that follows
linked: Resource data is at the link provided
attached: Resource data is in an attachment.

Clearly, many paradata documents may be associated with one learning resource. Multiple metadata descriptions of a learning resource may also be contributed. For example, using different metadata schemas, as authored by, or for, different communities, etc. What links these documents to a particular resource is the *resource locator* (typically a URI) in the resource data description. Thus, if one were interested in finding all the metadata and paradata associated with a particular learning resource, they would do so by finding resource data descriptions whose resource locator is that of the resource in question.

Document Updating and Lifetime

Generally, documents submitted to the Learning Registry may not be modified after submission. If an update is required, the publisher simply submits the new version of the document to the Learning Registry. When multiple versions are found (For instance, by identifying that documents have the same type, submitter, and resource locator) its up to the data consumer to identify the most recent.

Submitters of data may also specify a lifetime for the document. How lifetime is handled by the Learning Registry, if at all, is not yet specified as of the current version of the specification.

Resource Data Description Data Model

The resource data description is a JSON document. The elements of the document are described below in the following format:

Element	Description	Required	Immutable
element_name `data_type (encoding)`	`About the element.`	■ `True` ☐ `False` ▣ `Conditional`	■ `True` ☐ `False` ▣ `Conditional`

{				
	doc_type resource_data	The literal "resource_data"	■	■
	doc_version literal	The literal for the current LR Spec version. (e.g. "0.23.0")	■	■

� *General elements about the submission.*

doc_ID string	Unique document ID within scope of the LR.	■	□
resource_data_type string	Open (best practices) vocabulary ["paradata", "resource", "assertion", ...]	■	■
active Boolean	Is the resource data description document active?	■	■

� *Information about the submission, independent of the resource data.*

identity		Identity and curation	■	□
{	**submitter_type** string	Fixed vocabulary ["anonymous", "user", "agent"]	■	■
	submitter string	Identity of the submitter of the resource data.	■	■
	curator string	Identity of the curator of resource data description.	□	
	owner string	Identity of the owner of the resource.	□	
	signer string	Identity of key owner used to sign the submission.	□	
}				

� *Submission and distribution workflow information.*

submitter_timestamp string (time/date)	Submitter-created timestamp.	□	□
submitter_TTL string (time/date)	Submitter statement of TTL of validity of submission.	□	□
publishing_node string	Node_id of node where injected into the network.	■	□
node_timestamp string (time/date)	Timestamp of when received by the current node.	■	■
create_timestamp string (time/date)	Timestamp of when first published to the network.	■	■

TOS			■	□
{	**submission_TOS**	Agreed terms of service by submitter	■	□

	string			
	submission_attribution string	Attribution statement from submitter	☐	☐
}				

do_not_distribute string	System provided key-value pair	☐	☐
weight integer	Submitter assigned weight (strength)-100:100	☐	☐

digital_signature		Digital signature of the submission	☐	☐
{	**signature** string	Signature string	■	☐
	key_location ["string"]	Array of public key locations	■	☐
	signing_method string	Fixed vocabulary ["LR-PGP.1.0"]	■	☐
}				

� *Information about the resource, independent of the resource data.*

resource_locator string	Unique locator for the resource described. Shall resolve to a single unique resource. ● "http://www.url/data/1961.jpg"	■	☐
keys ["string1","string2",...]	Array of hashtag, keyword value list used for filtering. ● ["energy", "gas", "collision", "LOM", "kinetics", etc...]	☐	☐
resource_TTL integer	TTL from resource owner for the resource itself, in days.	☐	☐

� *Applicable if the submission is a resource.*

payload_placement string	Required: if the submission is a resource. Fixed vocabulary ["inline","linked","attached"] ○ "inline": resource data is in an object that follows ○ "linked": resource data is at the link provided ○ "attached": resource data is in an attachment	▣	
payload_schema ["string"]	Required: if the submission is a resource. Array of schema description and keywords for the resource data. May use any metadata standard. ○ ["LODE", "LOM", etc...]	▣	

payload_schema_locator **string**	Optional: if the submission is a resource. Otherwise ignored. **Schema locator for the resource data.** ○ "http://www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd"	☐	
payload_schema_format **string**	Optional: if the submission is a resource. Otherwise ignored. schema MIME type	☐	
payload_locator **string**	Required: if the submission is a resource AND payload_placement value is "linked" Otherwise ignored. ○ "http://www.url/files/filename.json"	▣	
resource_data <the resource data object>	Required: if the submission is a resource AND payload_placement value is "inline" Otherwise ignored. The actual inline resource data (resource, metadata, paradata). May be a JSON object, a string encoding XML or some other format, or a string encoding binary. ● "<expression xsi:schemaLocation=\"http://www.url/xsd/v1p0 http://www.url/lode/v1p0.xsd\" xmlns=\"http://www.url/v1p0\" xmlns:xsi=\"http://www.w3.org/2001/someschema1\"> <identifier><catalog>abc</catalog> <entry>123/135/entry> etc... <title> <string language=\"en\">Where is the dog?</string> <string language=\"de\">Wo ist der hund?</string> </title> etc... <entity>BEGIN:VCARD VERSION:3.0 FN:Iris Florian Children's Books N:Iris Florian ADR:1A Neuenstrasse, Nordsee, Germany END:VCARD</entity> etc..."	▣	

� *Extensibility.*

X_xxx string	Placeholder for extensibility	☐	■

}

Extensibility

The data model may be extended with additional optional, mutable elements that describe a resource and have a character string value space. These extended elements must have names that begin with "X_". Such elements should provide additional hints about the resource content which, while also in the full metadata description, allow faster filtering based on the hints rather than information at the resource data description document level. Examples include: "X_subject", "X_title", "X_format", etc.

Generating Resource Data Descriptions

Many elements are either fixed with a constant value, automatically generated, or consistent for a given user. Some examples are some in Figure 2.

Fixed Elements	`doc_type`	`doc_version`	`active`		
Automatically Generated Elements	`doc_ID`	`publishing_node`	`update_timestamp`	`node_timestamp`	`create_timestamp`
Consistent Elements	`submitter_type`	`submitter`	`submission_TOS`		
Min Req'd Properties Differing Between Submissions	`resource_data_type`	`resource_locator`	`payload_placement`	`payload_locator` (if payload is linked)	`resource_data` (if payload is inline)

Figure 2. Example Element Value Types.

Thus, beyond the boilerplate, the minimum submission to the Learning Registry states what kind of resource data is being submitted (metadata, paradata, or the resource itself), what resource is being described (via its locator, typically a URI), and where the resource data can be found (either a locator or inlined.)

The following elements also vary from submission to submission, but are optional: `submitter_timestamp`, `submitter_TTL`, `keys`, `resource_TTL`, `payload_schema`, `payload_schema_locator`.

Below, *Figure 3* shows an example of a Resource Data Description Document.

```
{   "doc_type": "resource_data",
    "resource_data": "Put_anything_like_metadata, xml_or_whatever_here",
    "keys": [
        "science",
        "what_ever_you_want"
    ],
    "TOS": {
        "submission_TOS": "http://www.learningregistry.org/tos/cc0/v0-5/"
    },
    "payload_placement": "inline",
    "resource_data_type": "metadata",
    "payload_schema": [
        "hashtags", "describing",
        "resource", "format"
    ],
    "doc_version": "0.23.0",
    "active": true,
    "resource_locator": "URI_of_resource",
    "identity": {
        "curator": "",
        "owner": "",
        "submitter": "Your name or organization here",
        "signer": "Your name or organization if signing the document",
        "submitter_type": "agent"
    }
}
```

Figure 3: Example Resource Data Description Document

Although not yet implemented, filtering and querying the "keys" and optional "X_" elements, while possibly redundant with the payload, help lead data consumers to your data. Filtering or querying properties of inlined metadata may also be considered for implementation.

Services

Learning Registry services and their APIs provide the functionality to push, discover, or pull resource data from the network. They are RESTful and accessed via HTTP, accepting arguments and delivering results in JSON notation.

In the following Learning Registry service descriptions the address of the services have been shortened for clarity. For example, if a service has an address of "http://lr.someregistry.edu/servicename", it's shortened simply to "/servicename".

A decentralized Learning Registry network is comprised of multiple nodes. There is no requirement that a node implement a service. However, if implemented it will be consistent with the service API. Refer to the complete Learning Registry Specification for additional information.

Publishing Services

Current push/publish services include:

Basic Publish
SWORD Publish

Basic Publish

<nodeURL>/publish

GET (Not Applicable)	POST <nodeURL>/publish

Basic Publish is the most basic, direct mechanism to publish resource data. The Basic Publish service pushes an instance of a the JSON resource data description document directly to a node in the network. It's performed via HTTP POST to <nodeURL>/publish.

GET

Arguments

POST

JSON Request Object

Contains the resource data description documents to be published.

Object	Type	Note
{		
"documents":[	array	Array of one or more resource data description documents.
{resource_data_description},		JSON Resource data description document(s).
{resource_data_description},		JSON Resource data description document(s).
etc...		etc...
]		
}		

Return Value

The future version is planned to include additional error codes with descriptive strings.

Code	String	Note
200	OK	The request has succeeded.
500	Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.

JSON Result Object

Object	Type	Note
{		
"OK":	Boolean	"true" if successful.
"error":	string	Only present if "OK" is "false". Contains description of error or failure.
"document_results":[{		Array containing a result entry for each document in the request.
"doc ID":	string	ID of the document if successful.
"OK":	Boolean	"true" if document was published successfully.
"error":	string	Only present if "OK" is "false". Contains description of error or failure.
}		

]		
}		

Notes

1. The service may be configured to return only document IDs and not full documents.
2. If the request document ID is not provided, the service may be configured to return all resource data description documents subset of the as determined by the service.
3. Returned documents are ordered by data with the most recent being first.

Sample Code

Examples have additional formatting for the purposes of readability.

Publish the JSON formatted file named "`fileName.json`".

cURL

Format	`curl -X POST -H "Content-Type:application/json" "<<url>>" -d @fileName.json`
Example	`curl -X POST -H "Content-Type:application/json" ↲ "http://testnode.org/publish ↲ -d @my_basicpublish_formatted.json"`
Return	`{"document_results":[ {"doc_ID": "klm174s08n05f031961td15sep011fad", "OK": "true"} ], "OK": "true" }`

SWORD Publish

<nodeURL>/swordpub

GET (Not Applicable)	**POST** <nodeURL>/swordpub

SWORD (*Simple Web-service Offering Repository Deposit*) Publish achieves the same end as Basic Publish, but does so using the SWORD protocol. The SWORD 1.3 API provides the mechanism for publishing to a node. A node corresponds to a single, particular SWORD collection. The service currently supports only JSON resource data description documents.

GET

Arguments

POST JSON Request Object

Contains the resource data to be published.

Object	Note
`{resource_data_description}`	A single resource data description document.

Return Value

The future version is planned to include additional error codes with descriptive strings.

Code	String	Note
200	OK	The request has succeeded.

500	Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.

XML Result Object

Notes

Sample Code

Examples have additional formatting for the purposes of readability.

Publish the JSON formatted file named "`fileName.json`".

cURL

Format	`curl -X POST -H "Content-Type:application/json" <<url>> -d @filename.json`

Example	`curl -X POST -H "Content-Type:application/json" ↲`
	`http://testnode.org/swordpub ↲`
	`-d @swordpublish.json`

Returns	

```
<?xml version="1.0"?>

   <entry xmlns="http://www.w3.org/2005/Atom" ↲
          xmlns:sword="http://purl.org/net/sword/">

     <title>klm174s08n05f031961td15sep011fad</title>
     <id>klm174s08n05f031961td15sep011fad</id>

     <updated>2011-10-17T17:58:59.061524Z</updated>

     <author><name>Learning Registry</name></author>
     <summary type="text">A summary</summary>

     <sword:userAgent>curl/7.21.7 (i386-pc-win32) libcurl/7.21.7 ↲
             OpenSSL/0.9.8r zlib/1.2.5 libidn/1.18 libssh2/1.2.8 ↲
             librtmp/2.3</sword:userAgent>

     <generator uri="http://testnode.org/sword" version="1.0"/>

     <content type="application/json" ↲
      src="http://testnode.org/obtain/klm174s08n05f031961td15sep011fad"/>


  </entry>
```

Retrieval Services

Services for pulling data from the Learning Registry include:

Basic Obtain
Basic Harvest
OAI-PMH Harvest

Basic Obtain is the simplest of the three retrieval services, providing the most direct way to retrieve documents from the Learning Registry. Basic Harvest provides more advanced services, and *Open Archives Initiative Protocol for Metadata Harvesting* (OAI-PMH) provides data access in a manner compliant with the OAI-PMH protocol.

Basic Obtain

<nodeURL>/obtain

GET <nodeURL>/obtain ?request_ID=<ID> &by_doc_ID=<true|false> &by_resource_ID=<true|false> &ids_only=<true|false> &resumption_token=<token>	**POST** <nodeURL>/obtain

The Basic Obtain service pulls an instance of a resource data description document (or a set of documents) directly from a node on a resource distribution network. It is the most basic, direct mechanism to access resource data.

Obtain has three modes:

by_doc_ID: If doc_IDs are included in a request, then the corresponding resource data description documents are returned.

by_resource_ID: If resource_locators are included in the request, then for each resource_locator an array of documents referring to that locator (whether they be metadata or paradata) are returned.

Not specified: If neither doc_IDs or resource_locators are supplied, the service may return all, or a service-determined subset, of the resource data description documents.

GET

Arguments

Object	Type	Note
`"by_doc_ID"`	`Boolean`	Optional: default"false". If "true" then request is for specific document for each ID in the list. request_ID value(s) are document IDs for a resource data description document(s).
`"by_resource_ID"`	`Boolean`	Optional: default "true". request_ID value(s) are unique resource identifier(s), e.g., the resource locator(s).
`"resumption_token"`	`string`	Optional: provided as a result of prior calls. Flow control resumption token.
`"ids_only"`	`Boolean`	Optional: default"false". If "true" the the request is just for IDs, not documents.
`"request_ID"`	`string`	Resource ID or resource data descoription document ID.

POST

JSON Request Object

Object	"true"ype	Note
{		List of resource IDs or resource data description document IDs to obtain.
"by_doc_ID":	Boolean	Optional: default"false". If "true" then request is for specific document for each ID in the list. request_ID value(s) are document IDs for a resource data description document(s).
"by_resource_ID":	Boolean	Optional: default "true". request_ID value(s) are unique resource identifier(s), e.g., the resource locator(s).
"resumption_token":	string	Optional: provided as a result of prior calls. Flow control resumption token.
"ids only":	Boolean	Optional: default"false". If "true" the the request is just for IDs, not documents.
"request_IDs":[1] [	array	Optional: ignored if ids_only is "true". Array of one or more resource IDs OR resource data description document IDs. If missing, documents are returned as determined by the service.
"request ID",	string	Resource IDs OR resource data description document IDs
"request_ID",		
etc...		
]		
}		

Return Value

The future version is planned to include additional error codes with descriptive strings.

Code	String	Note
200	OK	The request has succeeded.
500	Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.

JSON Result Object

Object	Type	Note
{		
"documents": [	array	Array of resource data description documents.[2]
{"document":	array	Array of JSON resource data description documents. Present only if: ○ Document ID is valid ○ Request was not for IDs only ○ otherwise NULL.
[{resource_data_description}]		JSON Resource data description document(s).

`,`		
`"doc_ID":`	`string`	
`},`		
`{"document":`		etc...
`[{resource_data_description} ]`		etc...
`etc...`		etc...
`}`		
`]`		
`"resumption_token":`	`string`	Flow control resumption token.[3] [4] Present only if: ○ flow control is supported ○ results require pagination. ○ NULL if this is the last set of paginated results
`}`		

Notes

1. If the request_ID is not provided, all resource data description documents, or a subset of documents may be returned as determined by the service.
2. The return value may be just document IDs and not full documents, as determined by the service.
3. Flow control implementation is determined by the service. If supported, the service determines how many documents to return per call. If the results returned is not the complete set the service returns one page of results and a resumption token. To retrieve the next page of results, the request would include the resumption token returned from the prior call.
4. The duration for which a resumption token is valid is determined by the service.

Sample Code

Examples have additional formatting for the purposes of readability.

Obtain a single document by its document ID. Specify that the request is for a document by setting `by_doc_ID` to "true".

cURL

Format	`curl -X GET "<<url>>/obtain ↲`
	`?by_doc_ID=true ↲`
	`&request_ID=some_Document_ID"`

Example	`curl -X GET "http://testnode.org/obtain ↲`
	`?by_doc_ID=true ↲`
	`&request_ID=klm174s08n05f031961td15sep011fad"`

Returns — ***Document Found***	`{`
	`  "documents": [{`
	`     "document":[{`
	`       "update_timestamp": "2011-09-07T20:27:59.747579Z",`
	`        etc.,...`
	`     }],`
	`     "doc_ID": "klm174s08n05f031961td15sep011fad", ...`
	`   }]`
	`}`

Returns — ***Document Not Found***	`{"documents":[], "resumption_token":null}`

Obtain a list of all resources returning only their IDs. The default value of `by_resource_ID` is "true".

cURL	***Format***	`curl -X GET "<<url>>/obtain ↲`
		`?ids_only=true"`

	Example	`curl -X GET "http://testnode.org/obtain ↲`
		`?ids_only=true"`

	Returns	`{"documents": [`
		`    {"doc_ID": "http://192.191.190.89/node/31"},`
		`    {"doc_ID": "http://192.191.190.19/node/28"},`
		`      etc...`
		`      etc...`
		`  ],`
		`  "resumption_token": ↲`
		`  "eyJhbGciOiAiSFMyNTYiLCAidHlwIjogIkpXVCJ9.eyJzdGFydGtleSI6ICJodHRwOi8vMTk`
		`  0Ljk1LjIwNy44OS9wcm9qZWt0ZS9wNzc5NjkuemlwIiwgImVuZGtleSI6IG51bGwsICJzdGFy`
		`  dGtleV9kb2NpZCI6ICJiOWM3N2ZkMDdmOTc0MzAxYWFkM2MxNzU0NmI1ODBlYyJ9.pBMEp5Dv`
		`  hYgfsrbiapGQIItevBpztWKiQBprwbzuE5Q"`
		`}`

The inclusion of "resumption_token" in the return value indicates that a single page of results was included and that another pager may be obtained by reissuing the request and including the "resumption_token" value.

Request another page of results.

cURL	***Format***	`curl X GET "<<url>>/obtain ↲`
		`?ids_only=true ↲`
		`&resumption_token=unique_token_string"`

	Example	`curl -X GET "http://testnode.org/obtain ↲`
		`?ids_only=true ↲`
		`&resumption_token= ↲`
		`eyJhbGciOiAiSFMyNTYiLCAidHlwIjogIkpXVCJ9.eyJzdGFydGtleSI6ICJodHRwOi8vMTk0`
		`Ljk1LjIwNy44OS9wcm9qZWt0ZS9wNzc5NjkuemlwIiwgImVuZGtleSI6IG51bGwsICJzdGFyd`
		`GtleV9kb2NpZCI6ICJiOWM3N2ZkMDdmOTc0MzAxYWFkM2MxNzU0NmI1ODBlYyJ9.pBMEp5Dvh`
		`YgfsrbiapGQIItevBpztWKiQBprwbzuE5Q"`

	Returns	`{"documents": [`
		`    {"doc_ID": "http://192.191.190.18/node/10"},`
		`    {"doc_ID": "http://192.191.190.18/node/69"},`
		`      etc...`
		`      etc...`
		`  ]`
		`}`

Basic Harvest

The Basic Harvest service may be used to connect to a node to harvest (pull) resource data description documents. Harvesting is done by resource data description document ID or by resource ID. Both GET and POST encoding of requests are supported.

The service is patterned after the OAI-PMH specification and is extensible so that it may fully support OAI-PMH–compliant harvesting. For additional information about the mapping of Learning Registry Basic Harvest to OAI-PMH consult the *Learning Registry Technical Specification*.

Basic Harvest includes six subservices:

GetRecord

ListRecords
Identify
ListIdentifiers
ListMetadataFormats
ListSets

Basic Harvest

<nodeURL>/harvest/getrecord

GET <nodeURL>/harvest/getrecord ?request_ID=<id> &by_doc_ID=<true|false> &by_resource_ID=<true|false>	**POST** <nodeURL>/harvest/getrecord

Getrecord returns the resource data description documents for a specified resource data document ID or resource ID. It operates similarly to Basic Obtain, with the principle exception that only a single doc_ID or resource_locator, rather than an array, may be specified in the request.[1] If the request ID is a unique resource identifier, such as the resource locator, the service returns all resource data description documents for the resource.

GET

Arguments

Object	Type	Note
`"request_ID":`	`string`	Resource ID or resource data description document ID to harvest.
`"by_doc_ID":`	`Boolean`	Optional: default "false". If "true" then request is for specific document for each ID in the list. request_ID value(s) are document IDs for a resource data description document(s).
`"by_resource_ID":`	`Boolean`	Optional: default "true". request_ID value(s) are unique resource identifier(s), e.g., the resource locator(s).

POST

JSON Request Object

Contains the resource data to be published.

Object	Type	Note
{		
`"request ID":`	`string`	Resource ID or resource data description document ID to harvest.
`"by_doc_ID":`	`Boolean`	Optional: default"false". If "true" then request is for specific document for each ID in the list.

		request_ID value(s) are document IDs for a resource data description document(s).
"by_resource_ID":	Boolean	Optional: default "true". request_ID value(s) are unique resource identifier(s), e.g., the resource locator(s).
}		

Return Value

The future version is planned to include additional error codes with descriptive strings.

Code	String	Note
200	OK	The request has succeeded.
500	Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.

JSON Result Object

Object	Type	Note
{		
"OK":	Boolean	"true" if successful
"error":	string	Description of error or failure. Present only if OK is"false".
"responseDate":	string	
"request":		The original API request.
{		
"verb":	string	The literal "getrecord".
"identifier":	string	Requested ID.
"by_doc_ID":	Boolean	
"by_resource_ID":	Boolean	
"HTTP request":	string	The HTTP request as a string.
}		
"getrecord": {		The requested resource data description document. Present only if request_ID is valid, otherwise NULL.
"record": [{		Record container
"header":{		Header info
"identifier":	string	Resource data description document ID.
"status":	string	Optional: "active" if not present. Fixed vocabulary ["active", "deleted"].
},		
"resource_data":		
{resource_data_description }		JSON Resource data description document.
}		
]		
}		
}		

Notes

1. Flow control is not currently supported.

Sample Code

Examples have additional formatting for the purposes of readability.

Get a single document by its document ID.

cURL

Format

```
curl -X GET "<<url>>/harvest/getrecord ↲
?request_ID=<<document ID>> ↲
&by_doc_ID=true"
```

Example

```
curl -X GET "http://testnode.org/harvest/getrecord ↲
?request_ID=klm174s08n05f031961td15sep011fad ↲
&by_doc_ID=true"
```

Returns

Document Found

```
{
  "OK": "true",
  "responseDate": "2011-09-28T14:27:08.598835Z",
  "error": ""

  "request": {
    "by_doc_ID": "true",
    "HTTP_request": "",
    "verb": "getrecord",
    "by_resource_ID": "false",
    "identifier": "klm174s08n05f031961td15sep011fad"
  },

  "getrecord": {
    "record": [{
        "header":{
          "datestamp": "2011-09-28T14:27:08.607941Z",
          "status": "active",
          "identifier": "klm174s08n05f031961td15sep011fad"
        },

        "resource_data": {
          etc...
          etc...
        }
    }]
  }
}
```

Returns

Document Not Found

```
{
  "OK": "false",
  "responseDate": "2011-09-28T19:58:17.240463Z",
  "error": "idDoesNotExist",
  "request": {
    "by_doc_ID": "true",
    "HTTP_request": "",
    "verb": "getrecord",
    "by_resource_ID": "false",
    "identifier": "the_invalid_request_ID_you_submitted"
  },
  "getrecord": {
  "record": []}
  }
}
```

Basic Harvest

<nodeURL>/harvest/listrecords

GET <nodeURL>/harvest/listrecords ?from=<date> &until=<date>	**POST** <nodeURL>/harvest/listrecords

Listrecords is used to retrieve resource data description documents submitted within a specific time/date range. The headers of returned records contain a document ID in the identifier field. There is currently no support for resource locators as identifiers.

GET

Arguments

Object	Type	Note
`"from"`	`string`	Optional. time/date[1] Per ISO 8601[2] Default is earliest timestamp Start of time/date range.
`"until"`	`string`	Optional. time/date[1] Per ISO 8601[2]. Default is latest timestamp End of time/date range.

POST

JSON Request Object

Contains the resource data to be published.

Object	Type	Note
`{`		
`"from":`	`string`	Optional. time/date[1] Per ISO 8601[2] Default is earliest timestamp Start of time/date range.
`"until":`	`string`	Optional. time/date[1] Per ISO 8601[2]. Default is latest timestamp End of time/date range.
`}`		

Return Value

The future version is planned to include additional error codes with descriptive strings.

Code	String	Note
200	OK	The request has succeeded.

500	Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.

JSON Result Object

Object	Type	Note
`{`		
`"OK":`	Boolean	
`"error":`	string	Descriptive string.
`"responseDate":`	string	Time/Date. Per ISO 8601[2] of this report.
`"request": {`		The original request.
`"verb":`	string	The literal "listrecords".
`"from":`	string	Time/Date. Per ISO 8601.[2]
`"until":`	string	Time/Date. Per ISO 8601.[2]
`"HTTP_request":`	string	The HTTP request as a string.
`},`		
`"listrecords": [{`	Array	Array of records.[1]
`"record": {`		
`"header": {`	string	
`"datestamp":`	string	Time/Date. Per ISO 8601[2] Resource data timestamp.
`"status":`	string	Optional: "active" if not present. Fixed vocabulary ["active", "deleted"].
`"identifier":`	string	Resource data description document ID.
`}`		
`}`		
`}]`		
`}`		

Notes

1. All matching results are returned in a single page as flow control is not currently supported.
2. Specified with one second accuracy per ISO 8601 *Data elements and interchange formats — Information interchange — Representation of dates and times*.

Sample Code

Examples have additional formatting for the purposes of readability.

Find all documents which were submitted on, or after, 1:00 am on the 23rd of September, 2011.

cURL

Format	`curl -o harvest_listrecords.json -X GET ↲ <<url>>/harvest/listrecords?from=<<date/time>>`
Example	`curl -o harvest_listrecords.json -X GET ↲ "http://testnode.org/harvest/listrecords ↲ ?from=2011-09-23T01:00:00Z`
Returns / ***Document Found***	see below

```
{"OK": true,
 "responseDate": "2011-09-29T17:10:01.830800Z",
 "error": "",
 "request": {
   "HTTP_request": "",
   "verb": "listrecords",
    "from": "2011-09-23T00:00:00Z"
  },
  "listrecords": [{
      "record": {
        "header": {
          "datestamp": "2011-09-29T17:10:02.074372Z",
          "status": "active",
          "identifier": "klm174s08n05f031961td15sep011fad"
        },
      "resource_data": {
        "doc_type": "resource_data",
         etc...
         etc...
      }},
    {"record":...},
    etc...
]}
```

Find all documents which were submitted on or between 01 Feb 2009, and 02 Feb 2009.

cURL

Format

```
curl -o harvest_listrecords.json -X GET ↲
<<url>>/harvest/listrecords?from=<<date/time>>&until=<<date/time>>
```

Example

```
curl -o harvest_listrecords.json -X GET ↲
"http://testnode.org/harvest/listrecords ↲
?from=2009-01-01T12:00:00Z&until=2009-01-02T12:00:00Z"
```

Returns

Document Not Found

```
{
  "request": {
     "HTTP_request": "",
     "verb": "listrecords",
     "from": "2009-02-01T12:00:00Z",
     "until": "2009-02-02T12:00:00Z"
  },
  "OK": true,
  "responseDate": "2011-09-29T20:31:03.785982Z",
  "error": ""
}
```