Learning Registry Technical Specification V 0.20.0

V 0.20.0 Archive -- Stable.

Technical Specification Introduction

This document provides the technical specification for the Learning Registry. It specifies the network model, data models and APIs. It does not specify policy-based operational procedures or the instantiation of the Learning Registry. It does not specify internal data models or internal APIs. The specification focuses on the APIs used by external agents and the data models and APIs required for overall operations of a network. While targeted at a document-oriented infrastructure, the specification itself is independent of any particular toolset. The document is currently a work in progress; its structure, organization and content are subject to change. All information is in this single monolithic document.

Learning Registry Overview

The Learning Registry [http://learningregistry.org] aims to make "learning resources easier to **find**, easier to **access** and easier to **integrate** into learning environments *wherever* they are stored -- around the country and the world." It defines a learning resource distribution network model and a set of open APIs and open interoperability standards to provide three fundamental, enabling capabilities:

1. a lightweight mechanism to publish (push) learning resources (or metadata or paradata describing the resources) into a learning resource distribution network, independent of format or data type (e.g., resource, metadata or paradata);
2. the ability for anyone to consume the published data and then, in turn, to publish additional feedback about the resources' use into the network (e.g., additional paradata), amplifying the overall knowledge about the resources;
3. a high-latency, loosely connected network of master-master synchronizing brokers distributing resources, metadata and paradata.

There is no central control, central registries or central repositories in the core resource distribution network. Published data can eventually flow to all nodes in the network. The network aims to be self assembling. Edge services can connect to any distribution node to find out what resources (and resource sources) are in the network, what's changed, what's being used, etc. Organizations may build consumer-facing, value-added services at the edge nodes to enable using, finding, sharing, and amplifying the resources, metadata and paradata for user communities. The Learning Registry provides *social networking for metadata* (trusted social collaboration around learning resources), enabling a *learning* *layer* on the social web.

Specification License

This specification is being developed under the Open Web Foundation Contributor License Agreement - Contributor Copyright Grant (CLA 0.9). The intent is that the final specification will be released under the Open Web Foundation Agreement (OWFa 0.9). Later versions may apply.

> Your use of this Specification may be subject to other third party rights. THIS SPECIFICATION IS PROVIDED “AS IS.” The contributors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to the Specification. The entire risk as to implementing or otherwise using the Specification is assumed by the Specification implementer and user. IN NO EVENT WILL ANY PARTY BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

The vocabulary of terms used in describing the Learning Registry are listed in the Glossary. Specific terms are set in **bold** when introduced in context.

Data models are described in a JSON-like notation. It follows JSON syntax, but instead of providing the value for a name, the data model defines the JavaScript data type of the named element. A description of the element, further restrictions on the value space (e.g., if a string is a URL) and if the element is optional or required is described in a comment. This model is used for convenience in early prototyping. A future version of the specification may describe the data models and their implementation binding independently.

Conformance

There is no overall conformance statement for this specification. The Learning Registry Test Suite (link TBD) MAY be used to test an implementation. However, passing the Test Suite does not imply conformance to this specification. There is no defined “reference implementation” (by definition when there is a conflict between this specification and the reference implementation, the reference implementation is considered to be authoritative).

All pseudo code is informative; it is not a normative implementation design. Behaviors defined in pseudo code are normative requirements on an implementation. Behaviors are usually defined in normative text.

An organization MAY place tighter requirements on an implementation than those stated, e.g., it MAY constrain a MAY, SHOULD or OPTIONAL clause to MUST, SHALL or REQUIRED. It MAY NOT relax any constraint.

Technical Specification Overview

The specification is split into several parts:

- **Network**: The description of the resource distribution network and its parts. A fixed multi-level structure of network parts is used to support distributing content and to provide policy-based security and operations.
- **Data Models**: The data models that are used to describe the network and learning resources data. Data models are document oriented.
- **APIs**: The APIs used to publish and consume data and those used to operate the network. The APIs are designed to abstract the logical behaviors of the Learning Registry from any particular implementation tools.
- **Operations**: Operational procedures that apply to any implementation.

Design Principles

The learning registry design and technical specification is based on several key principles:

- **Decentralized**: There are no centralized registries or repositories or central data stores. Thus all core data is replicated across the network.
- **Redundant**: There is no single point of failure in the design (an implementation may have single points of failure).
- **Abstracted**: Abstraction is critical to layering capabilities, e.g., network content replication is content type agnostic.
- **Minimal**: Specify only what is required. Features that are community specific or can be layered on top of the core design are excluded from the specification although essential elements needed to support such modeling are included.
- **Generic**: Prefer approaches, models, standards, etc., that have wide uptake beyond just the learning technology and digital repository space.
- **Secure**: Security is by design, e.g., default values lock down an implementation and must be explicitly overridden even to do common operations.
- **Trusted**: Data and operations need to be authentic and trusted
- **Document Oriented**: The design is targeted at a document-oriented system for implementation using document-oriented databases.
- **RESTful**: APIs are RESTful, and use CoolURIs to return different data representations.
- **Scalable**: The design needs to seamlessly scale and perform at scale.
- **Extensible and Enabling**: The design is meant to enable new capabilities. Unless explicitly restricted (usually to satisfy security requirements) anything in the design is extensible.

- **Web 2.0 Friendly**: The design is based on current, widely implemented Web 2.0 technologies.

Resource Distribution Network Model

The core of the Learning Registry is the network of loosely connected master-master synchronizing broker nodes distributing resources, metadata and paradata. Consumers and producers (edge node consumer and producer agents) connect to network nodes to inject information into the network or to extract information for external processing.

The network model is defined in terms of nodes, their services, the assembly of nodes into resource distribution networks, and the structuring of networks into communities. This two-tiered model of networks and communities supports security requirements for partitioning of resource data between different communities.

Network Nodes and Node Services

A **node** is a server process with network connectivity to either other nodes or to edge services. Nodes process **resource data** (e.g., network messages about resources, metadata, paradata, etc.).

A node SHALL be described using the network node data model. Only the owner of a node description MAY change the description. Certain attributes of the node description are immutable. *NB*: These are security constraints.

A node MAY provide five (5) different classes of services (APIs, services protocols and data models are detailed below) :

- **Publish Services**: Publish services are used by external agents to push (publish) resource data from the external agent into the distribution network. The data model for publication data is specified below. A node that provides publish services MAY support different publishing APIs, but all SHALL use the publication data model. The definitions of several common publishing APIs are specified below.
- **Access Services**: Access services are used by external agents to discover, access and obtain (pull) resource data from the distribution network. A node that provides access services MAY support different access APIs. The definitions of several common access APIs are specified below.
- **Distribution Services**: Distribution services are used to transfer, replicate and synchronize resource data from node X to node Y. X is the source node for distribution, Y is the destination node. To support security requirements, distribution is directed from X to Y; independent paired services [X→Y + Y→X] are used for bi-directional synchronization. The distribution services SHALL use the distribution data model specified below.
- **Broker Services**: Broker services operate at a node to augment, transform or process resource data held at that node to produce new or updated resource data for access or distribution. A node that provides broker services MAY support different broker processes. The definitions of several common broker processes are specified below.
- **Administrative Services**: Administrative services are used to query a node to obtain its status or

to trigger node administrative actions. The definitions of several common administrative services and APIs are specified below.

NB: There are no requirements to provision any service at a node. Provisioning requirements MAY be established by the policies of a particular network or community. This specification permits non operational or non accessible networks.

Network Topology

A **resource distribution network** is a group of one or more connected nodes, with each node providing node services. All nodes in a resource distribution network operate under the same policies. Multiple resource distribution networks MAY be established.

A resource distribution network SHALL be described using the resource distribution network data model. Only the owner of a network description MAY change the description. Certain attributes of the resource distribution network description are immutable. *NB*: These are security constraints.

Two types of network nodes and connectivity within a network are defined:

- **Common Node**: A common node MAY provide any of the node service classes listed. If provided, the distribution services of a common node SHALL be limited to connecting to other nodes in the same network (the distribution service MAY connect to multiple destination nodes). A common node is denoted CN herein.
- **Gateway Node**: A gateway node SHALL provide a distribution service. A gateway node MAY connect to one or more common nodes within the same network. A gateway node SHALL connect to and provide resource distribution to a gateway node in another network. A gateway node MAY provide administrative services. A gateway node SHALL NOT provide publish, access or broker services. A gateway node is denoted GN herein. *NB*: As defined, a gateway is a 1:1 interconnect between two networks. 1:1 is used to simplify topology in support of security requirements; it is not a technical constraint. *NB*: Multiple gateway nodes between two networks are permitted.

A node SHALL participate in, and be subject to the policies of, only one resource distribution network.

A node SHALL not transition or be moved from one resource distribution network to another. A node MAY only be added to or removed from a distribution network. *NB*: This is a security constraint.

A gateway node X that participates in some network N1 SHALL connect to a gateway node Y that participates in some other network N2. A gateway node SHALL NOT connect to any other nodes in network N2 or to any node in any other network.

A **network community** is a collection of interconnected resource distribution networks. A community MAY contain one or more resource distribution networks. A resource network SHALL be a member of only one community. Gateway nodes provide the connectivity between resources networks within a network community and MAY provide connectivity between networks in different communities. *NB*: A gateway node that provides an intra-community network gateway is undifferentiated from one that provides an inter-community network gateway.

A network community SHALL be described using the network community data model. Only the owner of a network community description MAY change the description. Certain attributes of the network community description are immutable. *NB*: These are security constraints.

Two types of network communities are defined:

- **Social Community**: A social community provides connectivity to other social communities. A network within a social community MAY connect to another network within the same social community or with a network that belongs to a different social community.
- **Closed Community**: A closed community provides no connectivity outside of the community. A network within a closed community SHALL NOT connect with another network within a different community.

For example, the Learning Registry is a social community; other social communities may connect to the Learning Registry community. For security and testing, the Learning Registry Testbed is a closed community, i.e., it consists of different networks (multiple networks to enable testing gateway protocols) but the testbed cannot be connected to the social production community.

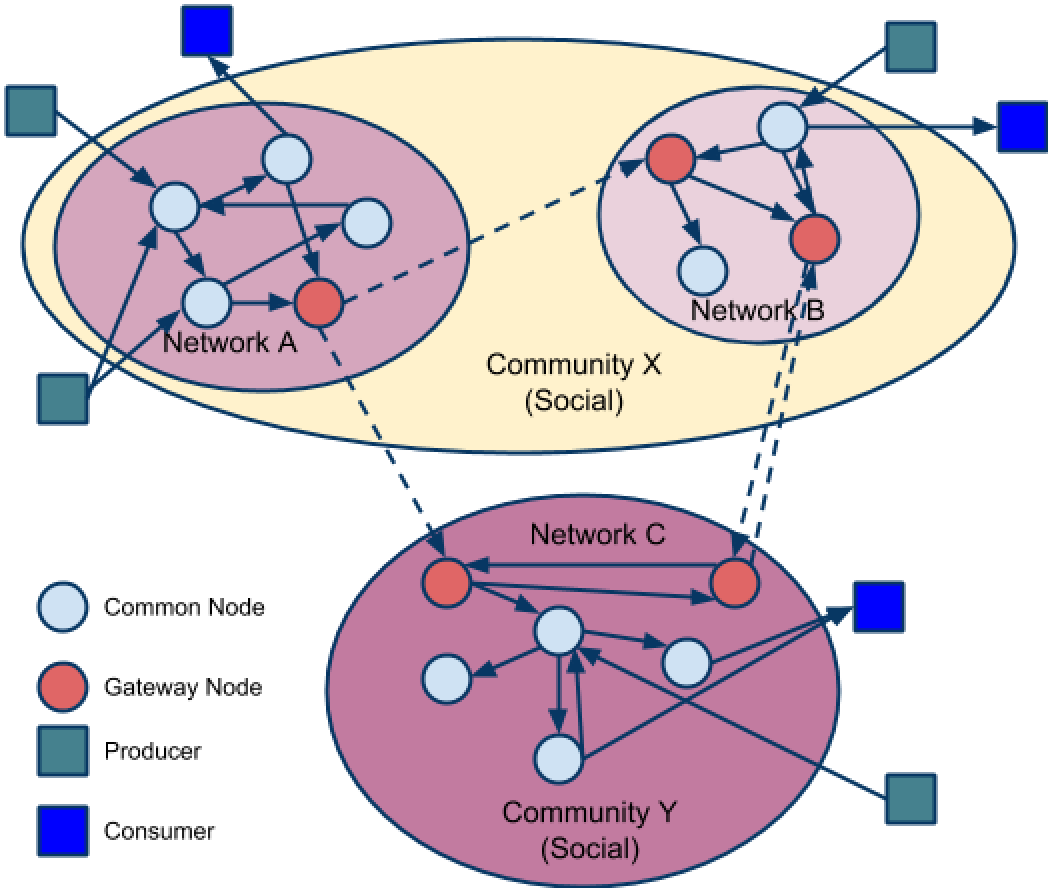
The Learning Registry community might consist of multiple networks and gateways. One network might be for uncurated OERs (open educational resources). A second network might be for curated OERs. And several others networks could be established for commercial resources (e.g., one per publisher). If the uncurated OER network has a gateway to the curated OER network, and there are gateways to each commercial networks, resource data can flow in only one direction, e.g., resource data for OERs into the commercial networks, but not the reverse.

A network SHALL not transition or be moved from one network community to another. A network MAY only be added to or removed from a network community. *NB*: This is a security constraint.

The resource network model provides nodes, collections of connected nodes within a network and the interconnection of networks in communities. The network model has this fixed hierarchy of components. Network communities connect to other communities using the same mechanism as networks that connect within a community.

Other network topology restrictions MAY be established by the policies of a particular network or community. This specification is intentionally minimal and does not define or limit other topologies, including degenerate topologies. *NB*: The model lets one design a network topology that might violate the policy and security constraints of a participating organization.

The diagram illustrates the network model. In the diagram there are three resource distribution networks (A, B, C) and two network communities (X and Y). Resource distribution network A connects to network B; both are part of the same community. Resource distribution network A also connects to network C and network C connects to network B. Resource distribution network C is in a different network community from A and B. If either network community X or Y was a closed community, the inter-network connection would not be permitted.



Network Data Models

The description of a network is maintained in a set of documents that includes:

- Network Node documents:
  - The description of the node.
  - The description of the connectivity of the node within the network (including gateways).
  - The description of the services provided by the node.
  - The description of the filters applied at a node.
- Resource Distribution Network documents:
  - The description of the resource distribution network that the node is a part of.
  - The description of the policies that govern the resource distribution network.
- Network Community documents:
  - The description of the network community that the node is a part of.

All data models MAY be extended with additional elements. The name of any extension element SHALL begin with the characters “X_” designating an extension element. Any document that includes any element that is not in the defined data model or is not an extension element is non conforming and

SHALL be rejected by any service.

All data models have a named attribute that is a “type” element (doc_type). The data model description specifies the literal value for this element for all instances of each type of document.

All data models have a named attribute that is a “version” element (doc_version). The data model description specifies the literal value for this element for all document instances.

All data models have a named attribute that indicates if the document instance is in use (active). Network data model document instances are never deleted; they transition from active to not active.

Additional constraints on attributes values are detailed below.

Network Node Description Data Model

The data model describing a node document. Once the data model has been instantiated for a node, the value of an immutable element SHALL NOT change. Other values MAY be changed only by the owner of the node document.

```
{
 "doc_type":              “node_description”,     // the literal “node_description”
                                                  // required, immutable
 ▲"doc_version":                  “0.20.0”,        // the literal for the current version -- “0.20.0”
                                           // required, immutable
 "doc_scope":             “node”,          // the literal “node”
                                           // required, immutable
 "active":                boolean,         // is the network node active
                                           // required, mutable from T to F only
 "node_id":               “string”,        // id of the node, required
                                           // unique within scope of the LR
                                           // immutable
 "node_name":             “string”,        // name of the node, optional
 "node_description":      “string”,        // description of the node, optional
▲ "node_admin_identity":         “string”,         // identity of node admin, optional
 "network_id":            “string”,        // id of the network that this node is a part of
                                           // recommended (required for gateway distribution)
                                           // immutable
"community_id":           “string”,        // id of the community that this node is a part of
                                           // recommended (required for gateway distribution)
                                           // immutable
"gateway_node":           boolean,         // T if node is a gateway node
                                           // recommended, F if not present, immutable
"open_connect_source":boolean,             // T if node is willing to be the source to
                                           // connect to any other node
                                           // F if node connectivity is restricted
                                           // recommended; F if not present
"open_connect_dest":      boolean,         // T if node is willing to be the destination
                                           //  to connect to any other node
```

```
                                            // F if node connectivity is restricted
                                            // recommended; F if not present
      “node_policy”:                        // node-specific policies, optional
       {
       "sync_frequency":        integer,    // target time between synchronizations in minutes
                                            // optional
        “deleted_data_policy”:  “string”,   // fixed vocabulary [“no”, “persistent”, “transient”]
                                            // see resource data persistence
      ▲ “TTL”:                  integer,    // minimum time to live for resource data in the node
                                            // in days, optional
                                            // overrides network policy TTL is larger than network
TTL
      ▲“accepted_TOS”:          [“string”]  ,       // list of ToS that the node will accept, optional
      ▲“accepts_anon”:          boolean,    // T if node is willing to take anonymous submissions
                                            // F if all submissions must be identified
                                            // optional, T if not present
      ▲“accepts_unsigned”:      boolean,    // T if node is willing to take unsigned submissions
                                            // F if all data must be signed
                                            // optional, T if not present
      ▲“validates_signature”:          boolean,     // T if node will validate signatures
                                            // F if node does not validate signatures
                                            // optional, F if not present
      ▲“check_trust”:           boolean     // T if node will evaluate trust of submitter
                                            // F if node does not check trust
                                            // optional, F if not present
        },
      ▲“node_key”:              “string”,   // node public key, optional
      "X_xxx":                  ?????       // placeholder for extensibility, optional
      }
```

▲*NB*: The node admin identity SHOULD be a URL, e.g., an email address. A deployment MAY specify that the identity be used to look up the node’s public key in a key server.

NB: Synchronization/replication frequency is maintained on a per node basis. This allows each node to sync on a different frequency (versus a network wide sync frequency), but does not allow each connection to a node to sync on a different frequency, which might complicate scheduling.

NB: The deleted data policy is used to support OAI-PMH harvest. It is part of the node description and not the service description since it controls overall node behavior and data persistence.

▲*NB*: The node MAY advertise its public key in the data model instance versus requiring key server lookup.

▲*NB*: The node MAY advertise its TTL. The value SHALL be ignored if it is smaller than the network policy TTL.

▲*NB*: The node MAY advertise the ToS that it will accept.

▲*NB*: A node MAY advertise that it does not accept anonymous submissions, e.g., resource data description documents where the submitter_type is anonymous. By default, anonymous submissions are supported.

▲*NB*: A node MAY advertise that it does not accept unsigned submissions. By default, submissions need not be signed.

▲*NB*: A node MAY advertise that it validates signatures. By default, all signatures are not validated.

▲*NB*: A node MAY advertise that it determines trust of submitter. By default, all trust is not checked.

▲*NB*: Signing, trust, etc., are all OPTIONAL. Default policy values imply that the node has a weak security and trust model. A node MUST explicitly state the policies it enforces.

Network Node Service Description Data Model

The data model describing a service description document; one document per service available at a node. Once the data model has been instantiated for a service, the value of an immutable element SHALL NOT change. Other values MAY be changed only by the owner of the node document.

NB: Ownership and control of the node description document and of the node service description document are vested in the same identity.

```
{
 "doc_type":               “service_description”,    // the literal “node_description”
                                                     // required, immutable
▲ "doc_version":                   “0.20.0”,          // the literal for the current version -- “0.20.0”
                                            // required, immutable
 "doc_scope":              “node”,          // the literal “node”
                                            // required, immutable
 "active":                 boolean,         // is the service active
                                            // required, mutable from T to F only
 "service_id":             “string”,        // id of the service, required
                                            // unique within scope of the LR
                                            // immutable
"service_type":            “string”,        // fixed vocabulary [“publish”, “access”,
                                            // “distribute”, “broker”, “administrative”]
                                            // required, immutable
"service_name":                    “string”,          // name of the service, optional
"service_description":     “string”,        // description of the service, optional
"service_version":         “string”,        // version number of the service description, required
                                            // version is local to the Learning Registry
                                            // not the version of some underlying spec for the service
"service_endpoint":        “string”,        // URL of service, required
▲”service_auth”:                            // service authentication and authorization descriptions
 {
▲"service_authz":          [“string”],      // fixed vocabulary
                                            // [“none”, “basicauth”, “oauth”, “ssh”,  ...]
                                            // required, mutable from ”none” to any stronger auth
▲“service_key”:            boolean,         // is a service key required to use the service
                                            // optional, immutable, default F
▲“service_https”:          boolean          // does service require https
                                            // optional, immutable, default F
 },
"service_data":            {},              // service-specific name-value pairs
                                            // optional
```

```
    "X_xxx":                ?????          // placeholder for extensibility, optional
    }
```

▲ ***Open Question*: What is the auth vocabulary?**

Network Node Connectivity Description Data Model

The data model describing a node connectivity document; one document per connection at a node. Once the data model has been instantiated for a connection, the value of an immutable element SHALL NOT change. Other values MAY be changed only by the owner of the node document.

NB: Ownership and control of the node description document and of the node connectivity description document are vested in the same identity.

```
    {
     "doc_type":             "connection_description",     // the literal "connection_description"
                                                           // required, immutable
     "doc_version":          "0.10.0",      // the literal for the current version -- "0.10.0"
                                            // required, immutable
     "doc_scope":            "node",        // the literal "node"
                                            // required, immutable
     "active":               boolean,       // is the connection active
                                            // required, mutable from T to F only
     "connection_id":        "string",      // id of the connection, required
                                            // unique within scope of the LR
                                            // immutable
    "source_node_url":       "string",      // URL of the source of the connection
                                            // required, immutable
    "destination_node_url":  "string",      // URL of the destination of the connection
                                            // required, immutable
    "gateway_connection":    boolean,       // T if this is a connection to a gateway node
                                            // F for a common node
                                            // recommended;  F if not present (common node)
                                            // immutable
    "X_xxx":                 ?????          // placeholder for extensibility, optional
    }
```

NB: By policy, there SHALL be only one document with a gateway_connection value of T per node.

NB: The source URL is not strictly needed. It is present to enable building a network map.

Working Assumption: It is assumed that a vocabulary to describe additional types of connections is not needed.

Network Node Filter Description Data Model

The data model describing a node filter; one document per node. Filters are used to restrict the resource data that is held at a node. Once the data model has been instantiated for a filter, the value of an immutable element SHALL NOT change. Other values MAY be changed only by the owner of the node document.

NB: Ownership and control of the node description document and of the node filter description document are vested in the same identity.

```
{
 "doc_type":            “filter_description”,    // the literal “filter_description”
                                                 // required, immutable
 "doc_version":         “0.10.0”,        // the literal for the current version -- “0.10.0”
                                         // required, immutable
 "doc_scope":           “node”,          // the literal “node”
                                         // required, immutable
 "active":              boolean,         // is the filter active
                                         // required, mutable from T to F only
“filter_name”:          “string”,        // name of the filter, optional
 “custom_filter”:       boolean,         // is this a custom filter (implemented in code, not rules)
                                         // required, if T, filter rules are ignored
“include_exclude”:      boolean,         // T if the filters describe what documents to accept
                                         // all others are rejected
                                         // F if the filters describe what documents to reject
                                         // all others are accepted
                                         // optional, T if not present
 “filter”:                               // array of filter rules
 [
  {“filter_key”:        “string”,        // REGEX that matches names in the
                                         // resource data description
                                         // required
    “filter_value”:     “string”         // REGEX that matches values in the
                                         // resource data description
                                         // optional, if not present, any value matches
  }
 ],
"X_xxx":                ?????            // placeholder for extensibility, optional
}
```

NB: Filters are optional.

NB: The same set of filters is applied in both the publication and distribution processes.

Resource Distribution Network Description Data Model

The data model describing a resource distribution network document. Once the data model has been instantiated for a network, the value of an immutable element SHALL NOT change. Other values MAY be changed only by the owner of the network description document.

```
{
 "doc_type":            “network_description”,   // the literal “network_description”
                                                 // required, immutable
▲ "doc_version":                “0.20.0”,        // the literal for the current version -- “0.20.0”
                                         // required, immutable
 "doc_scope":           “network”,       // the literal “network”
                                         // required, immutable
 "active":              boolean,         // is the resource distribution network active
                                         // required, mutable from T to F only
 "network_id":          “string”,        // id of the network, required
```

```
                                                  // unique within scope of the LR
                                                  // immutable
 "network_name":           “string”,              // name of the network, optional
 "network_description":    “string”,              // description of the network, optional
▲ "network_admin_identity":      “string”,              // identity of network admin, optional
 "community_id":           “string”,              // id of the community that this node is a part of
                                                  // recommended
                                                  // immutable
 ▲“network_key”:           “string”,              // network public key, optional
"X_xxx":                   ?????                  // placeholder for extensibility, optional
}
```

▲*NB*: The network admin identity SHOULD be a URL, e.g., an email address. A deployment MAY specify that the identity be used to look up the network’s public key in a key server.
▲*NB*: The network MAY advertise its public key in the data model instance versus requiring key server lookup.

Resource Distribution Network Policy Data Model

The data model describing the policies of a resource distribution network document. Once the data model has been instantiated for a network, the value of an immutable element SHALL NOT change. Other values MAY be changed only by the owner of the network *description* document.
NB: Ownership and control of the network description document and of the policy description document are vested in the same identity.

```
{
 "doc_type":               “policy_description”,      // the literal “policy_description”
                                                      // required, immutable
 "doc_version":            “0.10.0”,              // the literal for the current version -- “0.10.0”
                                                  // required, immutable
 "doc_scope":              “network”,             // the literal “network”
                                                  // required, immutable
 "active":                 boolean,               // are the policies active
                                                  // required, mutable from T to F only
 "network_id":             “string”,              // id of the network, required
                                                  // unique within scope of the LR
                                                  // immutable
 "policy_id":              “string”,               // id of the policy description, required
                                                  // unique within scope of the LR
                                                  // immutable
 "policy_version":         “string”,              // version identifier for the policy
  “TTL”:                   integer,               // minimum time to live for resource data in the network
                                                  // in days, required
 “policy_element_x”:       ?????,                 // placeholder for more policy elements
"X_xxx":                   ?????                  // placeholder for extensibility, optional
}
```

NB: The list of policy elements is currently incomplete.

Open Question: Should there be a node or network policy about anonymous submissions? ▲Set as node and network policy -- network default that node overrides.
Open Question: Should there be a node or network policy listing valid ToS? ▲Set as node policy.
Open Question: Should there be a node or network policy listing supported metadata and paradata schemata? ▲ No -- discoverable via OAI

Network Community Description Data Model

The data model describing a network community document. Once the data model has been instantiated for a community description, the value of an immutable element SHALL NOT change. Other values MAY be changed only by the owner of the network community description.

```
{
 "doc_type":              “community_description”,         // the literal “community_description”
                                                           // required, immutable
▲ "doc_version":                    “0.20.0”,          // the literal for the current version -- “0.20.0”
                                                  // required, immutable
 "doc_scope":             “community”,            // the literal “community”
                                                  // required, immutable
 "active":                boolean,                // is the network community active
                                                  // required, mutable from T to F only
 "community_id":          “string” ,              // id of the community, required
                                                  // unique within scope of the LR
                                                  // immutable
 “community_name”:        “string”,               // name of the community, optional
 “community_description:“string”,                 // description of the community, optional
▲ “community_admin_identity”:“string”,         // identity of community admin, optional
 “social_community”:      boolean,                // T if the community is a social community
                                                  // F if the community is a closed community
                                                  // recommended; F if not present (closed community)
                                                  // immutable
 ▲“community_key”:        “string”,               // node public key, optional
"X_xxx":                  ?????                   // placeholder for extensibility, optional
}
```

▲*NB*: The community admin identity SHOULD be a URL, e.g., an email address. A deployment MAY specify that the identity be used to look up the community’s public key in a key server.
▲*NB*: Policies are described at the node or network level, not the community level.

Network Description

A valid, consistent network SHALL be described through a set of documents stored at each node in the network.

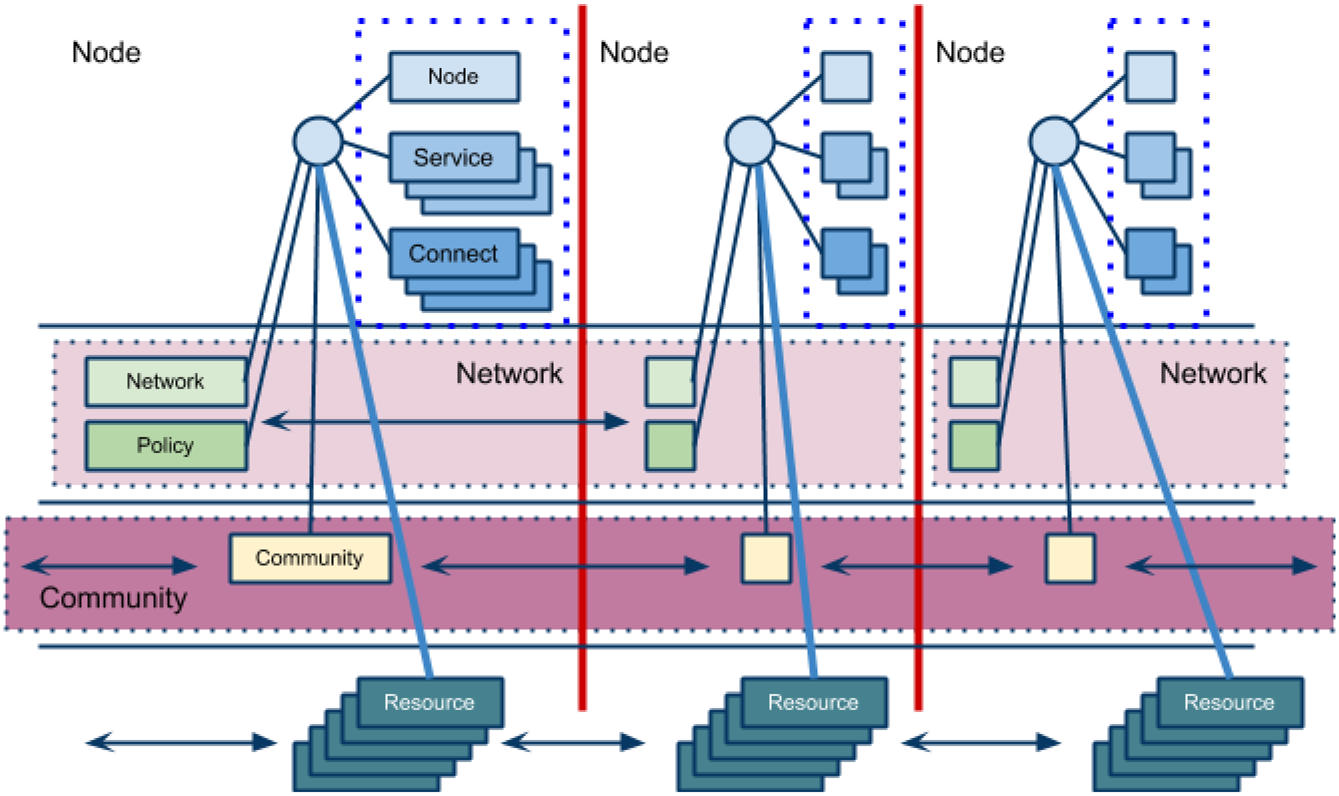
- Each node SHALL store one instance of the network node description document. A document SHALL be unique per node.
- Each node SHALL store one instance of the network node services document for each service that it provides. A document SHALL be unique per node.
- Each node SHALL store one instance of the network node connectivity document for each

connection to the node. A document SHALL be unique per node.

- Each node MAY store one instance of the network node filter document. A document SHALL be unique per node.
- Each node SHALL store one instance of the resource distribution network description document. This document SHALL describe the network that the node is a part of. The contents of this document SHALL be identical for all nodes in the network.
- Each node SHALL store one instance of the resource distribution network policy document. This document SHALL describe the policies of the network that the node is a part of. The contents of this document SHALL be identical for all nodes in the network.
- The node SHALL store one instance of the network community description document. This document SHALL describe the community that the network is a part of. The contents of this document SHALL be identical for all nodes in the community.

Additional types of node and network description documents MAY be defined, but SHALL be defined as either (1) unique per node, (2) identical for nodes in a network or (3) identical for all nodes in a community. Other organizational classifications SHALL NOT be used.

The illustration shows the mapping of documents to nodes and the distribution and synchronization of documents within resource distribution networks and network communities. *NB*: Filters are not shown.



Resource Data Data Models

The resource distribution network and its nodes process and distribute **resource data** (e.g., network messages about resources, metadata, paradata, etc.). Producer edge nodes publish resource data to a node of the network; the resource distribution network moves it to other nodes, and consumer edge nodes pull resource data for external use from nodes.

Resource data is described in a different documents that includes:

- The description of the resource, metadata, paradata, etc.

All data models MAY be extended with additional elements. The name of any extension element SHALL begin with the characters “X_” designating an extension element. Any document that includes any element that is not is the defined data model or is not an extension element is non conforming and SHALL be rejected by any service.

All data models have a named attribute that is a “type” element (doc_type). The data model description specifies the literal value for this element for all instances of each type of document.

All data models have a named attribute that is a “version” element (doc_version). The data model description specifies the literal value for this element for all document instances.

Additional constraints on attributes values are detailed below.

Resource Data Description Data Model

The data model describing resources, metadata, paradata, etc., that is distributed by the resource distribution network. The data model MAY be extended with additional optional, mutable elements that describe a resource and have a character string value space. The name of such an element SHALL begin with the characters “resource_”. Once the data model has been instantiated the value of an immutable element SHALL NOT change. Other values MAY be changed only by the owner of the document.

```
{
 "doc_type":             “resource_data”,          // the literal “resource_data”
                                          // required, immutable
 "doc_version":          ▲“0.20.0”,        // the literal for the current version -- “0.20.0”
                                          // required, immutable

 // general elements about the submission
 “doc_ID”:               “string”,         // unique ID for this resource data description document
                                          // unique within scope of the LR
                                          // immutable
                                          // user optional, required for storage
                                          // system generated when publishing
                                          // the document if not provided
 “resource_data_type”:   “string”,         // fixed vocabulary [“metadata”, “paradata”, “resource”]
                                          // required, immutable
 "active":               boolean,          // is the resource data description document active
                                          // required, mutable from T to F only
// information about the submission and flow, independent of the resource data
“submitter_type”:          “string”,           // fixed vocabulary [“anonymous”, “user”, “agent”]
                                               // required, immutable
                                               // “anonymous” -- submitter is unknown
                                               // “user” -- submitter is a user or has a user identity
                                               // “agent” -- submitter is an agent, e.g., a repository, LMS
“submitter”:               “string”,           // identity of the submitter of the resource data
                                               // required, immutable
                                               // use “anonymous” for type “anonymous”
“submitter_timestamp”: “string”,               // submitter-created timestamp
                                               // time/date encoding
                                               // optional
“submitter_TTL”:           “string”,           // submitter statement of TTL of validity of submission
                                               // time/date encoding
                                               // optional
“publishing_node”:         “string”,           // node_id of node where injected into the network
                                               // required
                                               // provided by the initial publish node (not distribution)
“update_timestamp”:        “string”,           // timestamp of when published to the network
                                               // of latest update
                                               // time/date encoding
                                               // required
                                               // provided by the initial publishing node
                                               // not by a distribution node
“node_timestamp”:          “string”,           // timestamp of when received by the current node
                                               // time/date encoding
                                               // optional
                                               // provided by the current distribution node
“create_timestamp”:        “string”,           // timestamp of when first published to the network
                                               // independent of updates
                                               // time/date encoding
                                               // required, immutable
                                               // provided by the initial publishing node on first publish
                                               // not by a distribution node or not an update
“TOS”: {                                       // terms of service
 “submission_TOS”:         “string”,           // agreed terms of service by submitter
                                               // required
“submission_attribution”:”string’              // attribution statement from submitter
}
▲“digital_signature”: {                        // digital signature of the submission, optional
 “signature”:              “string”,           // signature string, required
 “key_server”:             [“string”],         // array of key server identities, required
 “key_owner”:              “string”            // key owner identity, optional
 }

// 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
                                            // required
 “keys”:                  [“string”],       // array of hashtag, keyword value list used for filtering
                                            // optional
 “resource_TTL”:          “string”,         // TTL from resource owner for the resource itself
                                            // optional

 // the actual resource data description elements
 // these elements are optional as a block if the submission is a resource
 “payload_placement”:     “string”,         // 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
                                            // required
 “payload_schema”:        [“string”],       // array of  schema description/keywords
                                            // for the resource data
                                            // required
                                            // defined metadata schema values
                                            // defined paradata schema values
 “payload_schema_locator”:“string”,         // schema locator for the resource data
                                            // optional
 “payload_schema_format”:“string”,          // schema MIME type
                                            // optional
 “payload_locator”:       “string”,         // locator if payload_placement value  is “linked”
                                            // required if “linked”, otherwise ignored
 “resource_data”:                           // the actual inline resource data
 the resource data object,                  // the resource data itself (resource. metadata, paradata)
                                            // maybe a JSON object, or
                                            // a string encoding XML or some other format, or
                                            // a string encoding binary
                                            // required if “inline” otherwise ignored

 "X_xxx":                 ?????             // placeholder for extensibility, optional
}
```

Timestamp values for update_timestamp, node_timestamp, and create_timestamp SHALL be UTC 0.

Open Question: Is there a use case that requires create_timestamp?
Open Question: Is locator sufficient, or do we still need an ID for the resource?
NB: The doc_ID is not required when a user creates a resource data description document. If missing, it SHALL be provided by a publishing service when the document is first published.
NB: Need to agree on the conventions for valid submitter, owner and TOS strings.
NB: Separating owner from submitter enables 3rd party submissions.
▲*NB*: If the key owner is not included in the digital signature, the submitter is assumed to be the key

owner.
▲*NB*: Separating the key owner from the submitter enables 3rd party signing of submissions.
▲NB: The signing mechanism is described under identity.
NB: The supplied resource_locator SHALL be a unique ID within the scope of the Learning Registry and SHALL resolve to a single resource from which someone may uniquely access the resource. The resource_locator is used to correlate multiple resource data descriptions about a single resource. Thus the locator needs to be specific to the resource.
NB: A resource data description document contains only one set of resource data (metadata, paradata).
NB: To submit data only for a resource (no metadata or paradata), the payload attributes MAY be omitted.
NB: There is currently no mechanism to update the resource_* attributes without resubmitting the metadata or paradata about the resource. This is consistent with document-oriented transactional atomicity.
NB: There are no restrictions on the actual resource data. The format and encoding are based on the defined payload_schema and payload_schema_format values.
NB: Best practices of how to use the resource_* attributes and how to encode values are not provided.
NB: Some elements MAY be mapped to DC terms or LOM schema elements.

Resource Data Description	**Dublin Core**	**IEEE LOM**
resource_data_owner		3.3.2: LifeCycle.Contribute.Entity when 2.3.1: LifeCycle.Contribute.Role has a value of “Submitter.”
resource_description	dc:description	1.4: General.Description
resource_format	dc:format	4.1: Technical.Format
resource_language	dc:language	1.3: General.Language
resource_locator	dc:ID	1.1: General.Identifier or 4.3: Technical.Location
resource_owner	dc:publisher	2.3.2: LifeCycle.Contribute.Entity when 2.3.1:LifeCycle.Contribute.Role has a value of “Publisher.”
resource_rights	dc:rights	6.3: Rights.Description
resource_relationship	dc:resource refinement qualifier	7.1: Relation.Kind

resource_subject	dc:subject	1.5: General.Keyword
resource_title	dc:title	1.2: General.Title
resource_TTL		
resource_type	dc:type	5.2: Educational.LearningResourceType
related_resource	dc:resource	7.2.1: Relation.Identifier.Identifier

Metadata Formats

The metadata in a resource data description MAY be defined using any metadata standard. Metadata documents SHALL include the reference to the defining standard or schema. The following list of schema values SHALL be used to refer to common schemata. Implementations MAY extend this list.

Metadata Standard	payload_schema **value**
Dublin Core 1.1	“DC 1.1”
IEEE LOM 2002	“IEEE LOM 2002”
OAI-PMH Dublin Core	“oai_dc”

Attached and linked metadata SHALL include appropriate schema definitions and schema locators in the metadata file.

NB. There is currently no machine-readable list of schemata. Such a list could be defined in additional network description documents.

Inline metadata SHALL be encoded in JSON structure or as a single JSON string wrapping the entire metadata document.

JSON encodings of metadata schemata (primarily for inline resource data) will be provided in a future draft of the specification.

Paradata Formats

The paradata in a resource data description MAY be defined using any paradata standard. Paradata documents SHALL include the reference to the defining standard or schema. The following list of schema values SHALL be used to refer to common schemata. Implementations MAY extend this list.

Paradata Standard	payload_schema **value**

Attached and linked paradata SHALL include appropriate schema definitions and schema locators in the paradata file.

NB. There is currently no machine-readable list of schemata. Such a list could be defined in additional network description documents.

Inline paradata SHALL be encoded in JSON structure or as a single JSON string wrapping the entire paradata document.

JSON encodings of paradata schemata (primarily for inline resource data) will be provided in a future draft of the specification.

Resource Data

The resource data SHALL be maintained in a set of documents stored at each node in the network.

- Each node MAY store one or more instances of the resource data description documents. All document instances stored at a node SHALL be unique. A document MAY be replicated at many nodes.

Additional types of resource data documents (documents that differ in purpose from resource data description documents) MAY be defined, but SHALL be defined as unique per node. Other types of resource data description documents SHALL NOT be defined and SHALL NOT be replicated. Other organizational classifications SHALL NOT be used. *NB*: These constraints are meant to restrict placing resource data in multiple different databases.

Identity and Digital Signatures

▲This section is new in this version of the specification. The remainder is not highlighted.

Resource data description documents MAY be *signed* with a digital signatures. The signing and identity approach insures there can be no impostors. A persona (individual or organization) has a digital identity that can be used to sign a document. Thus two resource data description documents signed by the same identity are both from the same persona (assuming the signer has protected their private data), and the signature is non repudiable.

A document's digital signature provides the means to validate the authenticity of the signer's identity and the integrity of the signed document. The signature can only be used to verify that the signer controls their digital identify. It does not indicate that the document can be trusted or that the signer's digital identity maps to any real world identity. Trust and reputation are not provided by identity or digital signatures, but are enabled by them.

Digital signing and validation of resource data description documents is an OPTIONAL feature of the specification. A deployment of the Learning Registry MAY require documents be signed and validated. If a resource data description document is to be signed and validated, the following procedures SHALL be used to sign the document and verify the signature.

Signing a Resource Data Description Document

The controller of the identity (persona) that is used to sign the resource data description document MUST have a private/public PKI (public key infrastructure) key pair. A deployment of the Learning Registry SHALL specify the digital signature scheme, i.e., how to generate PKI keys and the encryption/signing model, e.g., X.509, OpenPGP/RFC 4048 with 1024bit DSA/ElGamal keys.

The controller of the identity MUST publish their public key at one or more key servers. Publication SHALL include the identity that can be used to look up the public key in the key server.

To sign the resource data description document, serialize part of JSON and apply the signing algorithm to the serialized JSON. The signature is applied to a UTF-8 text string from the concatenated set of values of the following key-value pairs in the order specified. All values SHALL be converted to text. Enclosing quotation marks on string values SHALL be removed before concatenation. Place a comma between values. Values in keys (an array of strings) SHALL be concatenated as individual items from the array, stripped of quotation marks and comma delimited. Any line break in a string SHALL be converted to a <CR><LF> character pair.

> "doc_type", "doc_version", “resource_data_type”, “active”, “submitter_type”, “submitter”, “submitter_timestamp”, “submitter_TTL”, “submission_TOS”, “submission_attribution”, “resource_locator”, “keys”, ““resource_TTL”, “payload_placement”

The serialized string is then *signed* (e.g., hashed, encrypted), using the signer’s private key yielding the value for the signature, i.e., a *detached signature*. The signer (key owner) MAY be an identity that is just used to sign the document, or it MAY be the identity of the submitter. Other identities SHALL NOT be used to sign the document.

NB: Only a portion of the document is signed. Since the document itself is not a literal string, the original representation MAY NOT be preserved when stored and transmitted.

Insert into the resource data description document the values for the digital_signature element.

- The signature, computed as described.
- The designation of one or more key_servers that can be used to obtain the public key of the signer. The designator of the key_server combined with the identity of the signer MUST be sufficient to obtain the public key. The value of a key_server designator SHALL be sufficient to identify the key server for public key lookup.
- An optional value of the key_owner as the identity of the signer if the submitter is not the signer. The submitter or signer identity is used by the key server to return the public key to verify

the signature. The value of key_owner SHALL be used when present; if not present, value of submitter SHALL be used in key lookup.

NB: This specification does not indicate how to obtain keys, sign documents or specify key servers. A deployment of the Learning Registry that requires digital signatures SHALL indicate the approach used for generating and publishing keys and signing documents.
NB: Currently only signing of resource data description documents is specified. A future version of the specification MAY require that other documents be signed. The signing process SHALL be the same; the elements of the data model used in the signature vary by document type.

Validating the Signature of a Resource Data Description Document

Any node or data consumer MAY validate a signature to determine if the signing party did sign the resource data description document.

Obtain the public key of the signer of the document. Use the value of key_owner if present, otherwise use the submitter value as the signer.

Iterate through the list of key_servers in the order provided. Look up the signer at the key server to obtain the signer's public key.

Serialize part of the JSON of the resource data description document. All values SHALL be UTF-8 encoding. Enclosing quotation marks on string values SHALL be removed before concatenation. Place a comma between values. Values in keys (an array of strings) SHALL be concatenated as individual items from the array, stripped of quotation marks and comma delimited. Any line break in a string SHALL be converted to a <CR><LF> character pair.

"doc_type", "doc_version", "resource_data_type", "active", "submitter_type", "submitter", "submitter_timestamp", "submitter_TTL", "submission_TOS", "submission_attribution", "resource_locator", "keys", "resource_TTL", "payload_placement"

NB: For the verification process to succeed, the serialized JSON must be an exact *lexical* match of the serialized JSON used to sign the document.

Using the signer's public key and the serialized JSON, verify the signature.

NB: Since only a portion of the resource data description document is used to compute the signature, there is no guarantee that the portion of the document that is not signed has not been tampered with.

Additional information on identity may be provided in a future version or draft of the specification.

Authorization and Authentication

▲This section is new in this version of the specification. The remainder is not highlighted.

Each service deployment at a node MAY specify authorization and authentication access controls and secure communications. These three types of controls are defined independently. Values for the controls are specified in the Network Node Service Description Model for the service. Full details of how services implement these controls is not specified.

Authentication

A service MAY require authentication to access the service. The service SHALL declare the authentication methods it supports. The service MAY support multiple authentication methods. Authentication methods are specified in the service_authz element of the service_auth element of the service description model.

The following authentication methods MAY be supported:

- None (none) -- the service is available without any authentication. If this authentication method is specified, other methods SHALL NOT be specified.
- Basic access authentication (basicauth) -- the service uses HTTP basic-auth for authentication. User identity and password credentials are included in the HTTP request. *NB*: A network node connectivity document includes the URL of the source and destination nodes used in content distribution, and a service description includes the URL of the service end point. For security, the URLs in these documents SHOULD NOT include credentials.
- OAUTH (oauth) -- the service is available through two-legged OAUTH.
- Secure Shell Protocol (SSH) -- the service is available through an SSH connection with SSH authentication.

NB: The list of authentication methods MAY be extended by a service.

Storage, processing and distribution of authentication credentials and establishing and provisioning OAUTH or SSH connections is out of scope for this specification. A deployment of the Learning Registry SHALL specify how to provision authentication.

Authorization

A service MAY require authorization to access the service. The service SHALL declare the authorization methods it supports.

Currently, only one authorization method is supported: a service MAY require an access key be included in the service request. A service that requires an access key specifies that the value of the service_key element of the service_auth element of the service description model is TRUE.

A service that uses an access key authorization SHALL include the access key in the HTTP header of the service call. What solution: Custom HTTP Header element, parameters, ...?

Storage, processing and distribution of access keys is out of scope for this specification. A deployment of the Learning Registry SHALL specify how to provision access keys.

Network Communications Security

A service MAY require that service HTTP requests be transmitted over a secure, encrypted communications channel. The service SHALL declare the network security methods it supports.

Currently only one network security method is supported: a service MAY require use of HTTPS. A service that requires a network security specifies that the value of the service_https element of the service_auth element of the service description model is TRUE.

Provisioning of HTTPS connections between clients and services is out of scope for this specification. A deployment of the Learning Registry SHALL specify how to provision secure communications.

Network Ports

Sevices may be access on specific TCP ports. The service_endpoint element of a service description and the source_node_url and destination_node_url elements of the network node connectivity document SHALL include port numbers.

Additional information on authorization and authentication may be provided in a future version or draft of the specification.

Trust

The section on security and information assurance will be provided in a future version or draft of the specification.

Security and Information Assurance

The section on security and information assurance will be provided in a future version or draft of the specification.

All services SHOULD maintain a secure log of all service actions. Details of logging requirements will be provided in a future version or draft of the specification.

Services and APIs

The services and their APIs provide the functionality that edge node producer and consumer agents use to push resource data into the distribution network and to discover and pull resource data from the network. They also define how to distribute the resource data throughout a network and how to manage and observe resource distribution network behavior.

The defined list of services follows. Any non gateway node MAY provide any of these services. A node MAY provide additional services not specified herein.

NB: There is no mechanism to reserve names for APIs, tag them as authoritative (i.e., they are defined in this specification), etc. A future version of the specification MAY extend the service definition to include tags (e.g., authoritative, experimental, third-party) and other validation or conformance information.

Services and APIs are RESTful and bound to a particular node in the resource distribution network. Service descriptions include the API call (HTTP binding), the API arguments, the message payload (using the JSON-like notation), the service results (using JSON-like notation), an informative pseudo code description of a possible implementation, and the network node service description data model.

The network node service description data model provides a machine and human readable description of the service; an instance of the description document is stored at the node that provides the service.

Additional constraints on API attributes, HTTP bindings (headers, HTTP errors), error processing and behaviors are described below.

Except as noted, services SHALL NOT be required to be provisioned at a node. An implementation SHALL NOT assume the provision of any service at any node, i.e., the implementation of one service cannot rely upon another service.

Resource Data Distribution Service

The resource data distribution service is used to distribute (synchronize or replicate) the resource data from one node to its connected nodes (unidirectional). The resource data distribution service SHALL apply filters, if present at the destination node, to restrict the resource data that is distributed. The resource data distribution service SHALL apply validation to restrict the resource data that is stored at the destination node. A destination node MAY reject any resource data description document from an unverified submitter or MAY reject resource data that does not conform to the node's ToS. The destination node MAY reject resource data that is not signed or where the signature cannot be verified.
NB: There is no defined mechanism to define the acceptable ToS for a node.
NB: How the resource data distribution service decides if it accepts or rejects resource data that does not come from a verified submitter, is not signed or that does not conform to the destination node's ToS is determined by the the node's policy and is not defined in this specification.

Future drafts or versions of this specification MAY define additional resource data distribution services.

The resource data distribution service is used in resource distrubution network operations. Access to the service SHOULD be restricted to network operational procedures.

A task scheduler SHOULD trigger resource data distribution from a source node to all of its destination nodes according to the frequency specified by the sync_frequency in the node description document.

The service provides three distinct functions:

- A service with an API at the source node which is used to trigger the resource data distribution.
- A service at each destination node that processes the incoming resource data.
- A service and API at each destination node that is used to return descriptive information from the

destination node to the source node to validate the connection.

API: Data Distribution

```
POST <nodeURL>/distribute

Arguments:
    None

Request Object:
    None

Results Object:                         // results summary and errors
    {
     "OK":       boolean,               // T if successful
     "error":    “string”               // text describing error
    }

Return Codes:
    200    Successful
    500    An error
```

Resource Distribution: Source Node Process

```
// Distribute a resource data description document collection from one node to its connected nodes
VALIDATE resource_document_database.doc_type = “resource_data”
// only distributing resource data
GET the network node description document for the source node to obtain
    source.network_id
    source.community_id
    source.gateway_node
GET the network community description document for the source node to obtain
    source.social_community
GET the network node connectivity documents for the source node
    IF > 1 connectivity document with gateway_connection = T
        THEN ABORT
        // only one gateway is allowed, faulty network description
    FOR EACH node connectivity document
        GET the network node description document for the destination node
        (via the destination_node_url) to obtain
            destination.network_id
            destination.community_id
            destination.gateway_node
        GET the network community description document for the destination node
        to obtain
            destination.social_community
        IF source.community_id <> destination.community_id
          AND ((NOT source,social_community)
          OR (NOT destination.social_community))
          THEN SKIP
// cannot distribute across non social communities
IF (NOT gateway_connection) AND
   source.network_id <> destination.network_id
          THEN SKIP
// cannot distribute across networks (or communities) unless gateway
IF gateway_connection AND
   source.network_id = destination.network_id
          THEN SKIP
// gateway must only distribute across different networks
IF gateway_connection
   AND ((NOT source.gateway_node)
          (OR (NOT destination.gateway_node))
          THEN SKIP
// gateways can only distribute to gateways
COMMIT all outstanding resource data description database operations
PERFORM distribution service between source and destination nodes
// this is the distribution primitive
```

NB: There may be a better way to do the validations via map-reduce.
NB: Since all attributes of the network that model its topology are immutable, the replication process should be transactionally safe.
NB: The process is only designed to distribute resource data. It encodes specific business rules about gateway processing. It SHOULD NOT be used to distribute network descriptions.
NB: The process does not return errors when there are bad connection descriptions; they are skipped. A bad connection should never have been accepted; the checks are included to ensure consistency.
NB: The process does not return errors when a distribution fails, either directly or because the destination node is not available. The process SHOULD verify that the destination node is reachable and operational before performing data distribution.

Resource Distribution: Destination Node Process

```
// Process and filter inbound resource data description documents at a node
VALIDATE the resource data description document
          // skip storing all documents that do not validate
REJECT the resource data description document if the submitter cannot be verified
REJECT the resource data description document if the submitter_TOS is unacceptable to the node
IF the network node filter description document exists and contains active filters
          THEN PERFORM filtering and store only documents that pass the filter
UPDATE node_timestamp // when the document was stored at the node
```

NB: The process does not return indicators when documents are filtered.
NB: An implementation SHALL maintain node_timestamp in a manner that does not trigger redistibution of the document; node_timestamp is a local node value.

API: Destination Node Information

```
GET <nodeURL>/destination

Arguments:
        None

Request Object:
        None

Results Object:                                 // results summary and errors
        {
         "OK":            boolean,              // T if successful
         "error":         “string”              // text describing error
         “target_node_info”:                    // target distribution node data
         {
          "active":               boolean;      // is the destination network node active
          “node_id”:              “string”,     // ID of the destination network node
          "network_id":           “string”,     // id of the network of the  destination
          "community_id":         “string” ,    // id of the community of the  destination
          "gateway_node":         boolean,      // destination node is a gateway node
          “social_community”:     boolean,      // is community a social community
        }

Return Codes:
        200     Successful
        500     An error
```

Resource Distribution: Destination Node Information

```
// Return the description of a destination network node
DEFINE VIEW on
        network node description document containing the required output fields
        + network community description document containing the required output fields
QUERY
RETURN Results document
```

Service Description

```
{
 "doc_type":            “service_description”,
 ▲"doc_version":                  “0.20.0”,
 "doc_scope":           “node”,
 "active":              true,
 "service_id":          “<uniqueid>”,
 "service_type":        “distribute”,
 "service_name":        “Resource Data Distribution”,
 "service_version":     “0.10.0”,
 "service_endpoint":    “<nodeURL>”,
▲”service_auth”:                       // service authentication and authorization descriptions
```

```
 {
▲"service_authz":      ["xxxx"],                  // authz values for the service
▲“service_key”:        T/F,            // does service use an access key
▲“service_https”:      T/F             // does service require https
}
}
```

The service definition is only for the source node. There is no service definition for the target node API used to get the target node information.

Resource Data Publish Services

Publish services are used to push resource data into the network. They are used by external publishing edge nodes. All resource data publishing services SHALL apply filters if present to restrict the resource data that is published to the node. All resource data publishing services SHALL apply validation to restrict the resource data that is published to the node. The validation process MAY also provide local updates to the resource document prior to it being published. ▲Any resource data publishing service MAY reject any resource data for any reason:

- From an untrusted submitter
- From an anonymous submitter
- Not signed
- Signature not valid
- Does not conform to the node’s ToS.

NB: There is no defined mechanism to define the acceptable ToS for a node. ▲A node MAY advertise acceptable ToS in the node description document, but this MAY not be accurate.
▲*NB*: How a data publishing service decides if it accepts or rejects resource data that comes from an untrusted submitter, is not signed, signature cannot be validated, or that does not conform to the data publishing service’s ToS is determined by the data publishing service’s policy and is not defined in this specification.

Future drafts or versions of this specification MAY define additional resource data publish services.

Basic Publish Service

The basic publish service pushes an instance of a resource data description document (or a set of documents) directly to a node in a resource distribution network. It is the most basic, direct mechanism to publish resource data.

Each resource data description document in the supplied set is published independently. In addition to the overall service return indicating status, there SHALL be one returned object per resource data description document, aligned 1:1 to the documents in the supplied resource data description document array, indicating status of publishing of the resource data description document.

Each resource data description document SHALL be published to the node’s resource data description document database. Prior to being published, it SHALL be validated: e.g., the syntax MUST be correct, mandatory values MUST be present, all values MUST come from the appropriate data space. The document SHALL also be subject to all filters defined at the node. Documents that do not pass the filters SHALL NOT be published. ▲The document MAY also be subject to verification of ToS and submitter information (including presence and validity of digital signature). Documents from anonymous submitters, untrusted submitters, unsigned documents, or documents with a ToS that is not acceptable to the node MAY be rejected.

The publication process provides values for specific elements in the resource data description document.

If the resource data description document does not have an assigned identifier, the service SHALL assign one and return the value.

If the resource data description document has an identifier and a document with the same identifier exists in the resource data description document collection, the new document SHALL be an update, replacing the existing document in total. If the resource data description document is being updated, the value of an immutable element SHALL NOT be changed.

The publication process SHALL set values for publish_node, update_timestamp, node_timestamp, create_timestamp. All timestamp values SHALL be the identical. All timestamp values SHALL be UTC 0.

NB: There are no restrictions on the size of a batch publish document set, either in the number of elements or the total size of the HTTP message. An implementation SHALL indicate any size limits in the service description.
NB: The process currently does not handle attachments.

Open Question: Publishing to the node is by the node owner. Do we need more to support trust?
NB: The process currently does not handle attachments.

API

```
    POST <nodeURL>/publish

    Arguments:
        None

    Request Object:                              // resource data to be published
        {"documents":
         [                                       // array of
          {resource_data_description}            // resource data description documents
        ]}

    Results Object:
        {
         "OK":          boolean,                 // T if successful
```

```
 "error":       "string",              // text describing global error
                                       // present only if NOT OK
"document_results":
[                                      // array of per document results
 "doc_ID":      "string",              // ID of the document
 "OK":          boolean                // T if document was published
 "error":       "string"               // text describing error or filter failure
                                       // present only if NOT OK
 ]
}
```

Return Codes:

200

500

Basic Publish

```
// Publish each resource data description document in the supplied list
VALIDATE the publish request // apply appropriate business rules
    IF there is an overall error
        THEN   // create the global error object
            OK := F
            error := "error msg" // an appropriate error for global condition
            EXIT
OK := T // global return status
FOR EACH resource data description document
    VALIDATE the resource data description document // all syntactical and semantic rules
    IF there is an error
        THEN // create an error object array element object for the individual document
            OK := F
            error := "error msg" // an appropriate error for the document
            doc_ID := supplied doc_ID
            SKIP
    IF the network node filter description document exists and contains active filters
        THEN PERFORM filtering and store only documents that pass the filter
        IF the resource data description document does NOT pass the filter
            THEN // indicate filtering was applied
                OK := F
                error := "rejected by filter" // an appropriate filtering message
                doc_ID := supplied doc_ID
                SKIP
    ▲IF the service applies ToS checks
        AND the resource data description document TOS is unacceptable
            THEN // indicate ToS was rejected
                OK := F
                error := "rejected by ToS" // an appropriate message
                doc_ID := supplied doc_ID
                SKIP
```

```
▲IF the service does not accept anonymous submissions
    AND the resource data description document has submitted_type==”anonymous”
        THEN // indicate submitted type was rejected
            OK := F
            error := “anon submission rejected” // an appropriate message
            doc_ID := supplied doc_ID
            SKIP
▲IF the service validates the submitter
    AND the resource data description document submitter cannot be verified or trusted
            THEN // indicate submitter was rejected
            OK := F
            error := “rejected submitter” // an appropriate message
            doc_ID := supplied doc_ID
            SKIP
▲IF the service requires a signature
    AND the resource data description document signature not present
            THEN // indicate signature was rejected
            OK := F
            error := “no signature” // an appropriate message
            doc_ID := supplied doc_ID
            SKIP
 ▲IF the service validates the signature
    AND the resource data description document signature cannot be verified
            THEN // indicate signature was rejected
            OK := F
            error := “rejected signature” // an appropriate message
            doc_ID := supplied doc_ID
            SKIP
IF resource data description document did not have a supplied doc_ID
    THEN generate a new unique doc_ID
PUBLISH the resource data description document to the node
    by the owner of the node
    to the node’s resource data description document database
    SET publish_node, update_timestamp, node_timestamp, create_timestamp
IF there is a publishing error
    THEN // create an error object array element object for the individual document
        OK := F
        error := “publish failed” // an appropriate error for the publish failure
        doc_ID := supplied doc_ID
        SKIP
// create a return object array element object for the individual document
OK := T
doc_ID // supplied or generated doc_ID
```

Service Description

```
    {
          "doc_type":               “service_description”,
        ▲ "doc_version":                     “0.20.0”,
          "doc_scope":              “node”,
          "active":                 true,
          "service_id":             “<uniqueid>”,
          "service_type":           “publish”,
          "service_name":           “Basic Publish”,
          "service_version":        “0.10.0”,
          "service_endpoint":       “<nodeURL>”,
        ▲”service_auth”:                          // service authentication and authorization descriptions
          {
        ▲"service_authz":          [“xxxx”],               // authz values for the service
        ▲“service_key”:            T/F,           // does service use an access key
        ▲“service_https”:          T/F            // does service require https
          }
          "service_data":
           {
            “doc_limit”:            integer,               // specify the maximum number of documents in
a batch
            “msg_size_limit”:       integer        // specify the maximum message size
           }
          }
```

SWORD Publish Service

SWORD (Simple Web-service Offering Repository Deposit) is a profile of the Atom Publishing Protocol (known as APP or ATOMPUB). The SWORD APP API provides a mechanism for a repository to publish its metadata or paradata to a node in the resource distribution network. Unless specified, the service SHALL support the SWORD V 1.3 protocol.

The SWORD service currently supports publishing of a resource data description document to a node. A node corresponds to a SWORD collection; there is only one collection to deposit to. The service supports SWORD developer features and mediated deposit. The service currently only supports the deposit JSON encoded resource data description documents. Package support is currently not specified.

The service end points for the protocol operations are:

Atom Pub Protocol Operation	**SWORD API Endpoint**
Retrieving a Service Document	GET <nodeurl>/swordservice
Listing Collections	Currently not supported. To be added in a later version of the specification.
Creating a Resource	POST <nodeurl>/swordpub

Editing a Resource	Currently not supported. May be added in a later version of the specification.
Deleting a Resource	Currently not supported. May be added in a later version of the specification
Retrieving a Resource	Not supported -- provided via the Atom Pub Service

Open Question: Should SWORD just publish the raw metadata or paradata document and let the service create the JSON?

Each of the protocol operations are specified separately. The Service Description document SHALL apply to the entire API.

Retrieve Service Document

The SWORD Service Document endpoint SHALL return an XML SWORD Service Document with the following settings:

- Global element settings:
  - <sword:version> element: 1.3
  - <sword:verbose> element: true
  - <sword:noOp> element: true
- Workspace settings: There SHALL be only one workspace. The <title> element of the workspace SHALL be the community_name from the *network community description data model*. If the community_name is missing, the value SHALL be the community_id from the *network community description data model.*
- Collection settings: There SHALL be only one collection.
  - IRI (http attribute): URL of the network node
  - <title> element: The title of the collection SHALL be the node_name from the *network node description data model*. If the node_name is missing, the value SHALL be the node_id from the *network node description data model*.
  - <accept> element: application/json
  - <sword:mediation> element: true
  - <dcterms:abstract> element: The abstract SHALL be the node_description from the *network node description data model*. If the node_description is missing, the element SHALL be omitted.
  - <sword:collectionPolicy> element MAY be present. The value is determined by the policies of the node, network or community (e.g., for the public Learning Registry community, the policy is the terms of service for the community, http://www.learningregisrty.org/tos/)
  - <sword:treatment> and <sword:service> elements SHALL be omitted.

API

```
GET <nodeurl>/swordservice

HTTP Headers
        X-On-Behalf-Of: [on-behalf-of-user]

Results XML
        Well formed XML instance document that conforms to the SWORD 1.3 specification.
                <?xml version="1.0" encoding='utf-8'?>
                <service xmlns="http://www.w3.org/2007/app"
                        xmlns:atom="http://www.w3.org/2005/Atom"
                        xmlns:sword="http://purl.org/net/sword/"
                        xmlns:dcterms="http://purl.org/dc/terms/">
                <sword:version>1.3</sword:version>
                <sword:verbose>true</sword:verbose>
                <sword:noOp>true</sword:verbose>
                  <workspace>
                        <atom:title>...</atom:title>
                        <collection href="...” >
                          <atom:title>...</atom:title>
                          <accept>application/json</accept>
                          <sword:mediation>true</sword:mediation>
                          <dcterms:abstract>...</dcterms:abstract>
                          <sword:collectionPolicy>...</sword:collectionPolicy>
                        </collection>
                  </workspace>
                </service>
```

SWORD: swordservice

```
        // return the service document
        Build XML results document
        EMIT the Atom Pub and SWORD namespace declarations
        EMIT the required elements
                <sword:version>1.3</sword:version>
                <sword:verbose>true</sword:verbose>
                <sword:noOp>true</sword:verbose>
        EMIT the workspace elements
                <workspace>
                        <atom:title>community_name or community_id from the network community
                        description data model<atom:title>
        IF the [on-behalf-of-user] is permitted to publish to the node
                THEN EMIT the collection elements
                        <collection
                                href=”URL of the network node”>
                                <atom:title>node_name or node_id from the network node
                                description data model</atom:title>
                                  <accept>application/json</accept>
                                  <sword:mediation>true</sword:mediation>
```

```
                    <dcterms:abstract>node_description from the network node
                        description data model</dcterms:abstract>
                  <sword:collectionPolicy>Policy URL</sword:collectionPolicy>
                  </collection>
Complete XML elements
      </workspace>
      </service>
```

Create a Resource

in a future draft of the specification

API

POST <nodeurl>/swordpub

HTTP Headers

Results XML

SWORD: swordpub

// pseudo code

Service Description

```
{
 "doc_type":              "service_description",
▲ "doc_version":                    "0.20.0",
 "doc_scope":             "node",
 "active":                true,
 "service_id":            "<uniqueid>",
 "service_type":          "publish",
 "service_name":          "SWORD APP Publish V1.3",
 "service_version":       "0.10.0",
 "service_endpoint":      "<nodeURL>",
▲"service_auth":                           // service authentication and authorization descriptions
 {
▲"service_authz":         ["xxxx"],                // authz values for the service
▲"service_key":           T/F,             // does service use an access key
▲"service_https":         T/F              // does service require https
 }
 "service_data":
  {
   "version":             "1.3"
   }
}
```

Basic Delete Service

The basic delete service "deletes" an instance of a resource data description document (or a set of documents) directly from a node in a resource distribution network.

Each resource data description document identified in the supplied set is deleted independently. In addition to the overall service return indicating status, there SHALL be one returned object per resource data description document, aligned 1:1 to the documents identified in the supplied resource data description document array, indicating deletion of the resource data description document.

The service MAY implement different deletion policies:

- *ignore* -- the deletion SHALL be acknowledged but the document is not deleted.
- *mark* -- the status of the document is changed to indicate that it has been deleted. The document SHALL NOT be returned by any access service.
- *delete* -- the document SHALL be deleted. What "deleted" means is dependent on the underlying implementation.
- *purge* -- the service SHALL, at some point, remove deleted documents.

NB: There are no restrictions on the size of a batch publish document set, either in the number of elements or the total size of the HTTP message. An implementation SHALL indicate any size limits in the service description.
NB: Only the owner of a document may delete it.
NB: A mechanism to delete all resource data description documents associated with a single resource identifier (resource locator) is not provided since these resource data description documents may have different owners.
NB: The deletion process SHALL be consistent with the resource data persistence policy.

API

```
        POST <nodeURL>/delete

        Arguments:
                None

        Request Object:                                  // list of resource data descriptions to delete
                {“request_IDs”:
                [                                        // array of
                 “doc_ID”:        ID                     // resource data description document ID
                                                         // required
                ]}

        Results Object:
                        {
                         "OK":             boolean,               // T if successful
                         "error":          “string”,              // text describing global error
                                                                  // present only if NOT OK
                        “document_results”:
                        [                                         // array of per document results
                         “doc_ID”:         “string”,              // ID of the document
```

```
    "OK":          boolean              // T if document was deleted
    "error":       “string”             // text describing deletion error
                                        // present only if NOT OK
    ]
   }
```

Return Codes:

```
    200
    500
```

Basic Delete

```
// Obtain the resource data description document for each supplied ID
FOR EACH resource data description document ID
    Put the resource data description document ID in the results object
    IF the document does not exist
        THEN
            OK := FALSE
            error := “document doesn’t exist”
            SKIP
    IF the document has been deleted
        THEN
            OK := FALSE
            error := “document already deleted
            SKIP
    // otherwise delete
    OK := TRUE
    CASE delete_action
        ignore:
            NO OP
        mark:
            set a flag on the document that it is deleted // ACTIVE := FALSE
        delete:
            perform a system-level delete // whatever “delete” means
        purge:
            perform a system-level delete // whatever “delete” means
            trigger system level purge // may run at some later time
```

Service Description

```
{
"doc_type":           “service_description”,
▲"doc_version":                  “0.20.0”,
"doc_scope":          “node”,
"active":             true,
"service_id":         “<uniqueid>”,
"service_type":       “delete”,
"service_name":       “Basic Delete”,
"service_version":    “0.10.0”,
    "service_endpoint":      "<nodeURL>",
   ▲"service_auth":                         // service authentication and authorization descriptions
    {
   ▲"service_authz":        ["xxxx"],              // authz values for the service
   ▲"service_key":          T/F,                // does service use an access key
   ▲"service_https":        T/F                 // does service require https
    }
    "service_data":
    {
      "delete_action":        "string",         // fixed vocabulary ["ignore", "mark", "delete", "purge"]
                                               // ignore -- ignore the delete request
                                               // mark -- mark the document as deleted
                                               // delete -- delete the document from the document store
                                               // purge -- purge the document
      "doc_limit":            integer,                 // specify the maximum number of documents in
a batch
      "msg_size_limit":       integer          // specify the maximum message size
      }
    }
```

Resource Data Access Services

Access services are used to pull resource data from the network. They are used by external access edge nodes to obtain one or more resource data description documents for "off network" processing. These services MAY be used to access individual resource data description documents by document ID or collations of all resource data description documents for each unique resource. Future drafts or versions of this specification MAY define additional resource data access services.

NB: The services do not currently define a specific process to find or maintain all resource data description documents in a collation for a specific resource locator, i.e., for a unique resource identifier. A future version of the specification MAY define additional resource data document types that maintain collation descriptions.

Basic Obtain Service

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.

For the list of supplied request IDs, the service SHALL return the corresponding resource data description documents from the node's resource data description document database where they exist. Optionally, the service MAY return just the document IDs and not full documents. The results SHALL be aligned 1:1 with the IDs in the request.

▲If the request ID is not provided, the service SHALL MAY return all or a service-determined subset of the resource data description documents. The service description SHALL specify how the service responds to an ALL request (returning ALL, none, or a limited subset). When returning a subset of the documents, the service SHOULD return the documents with the most recent node_timestamp values. *NB*: To support buffering, the service MAY return a smaller number of results than it advertises. Open Question: What does most recent mean when requesting via resource ID.

Request IDs MAY be:

- the document ID for a resource data description document. The service SHALL return the single resource data description document that matches the ID.
- a unique resource identifier, e.g., the resource locator. The service SHALL return all resource data description documents for the specified resource.

If a specified ID does no resolve to a resource data description document in the node's resource data description document database, the result object returned SHALL be NULL.
▲The internal storage of a resource data description document MAY include additional key-value pairs not defined in this specification. The service MAY return all stored key-value pairs, or only the key-value pairs defined in this specification for the resource data description document. The service document SHALL indicate if the returned documents are limited to the specification-defined key-value pairs only or if all stored key-value pairs are returned.

NB: There are no restrictions on number of requested documents or in the total size of the HTTP message or response. An implementation SHALL indicate any size limits in the service description.
NB: The default is that IDs are for resources, not documents.
NB: The default is to return full resource data description documents, not just IDs.
NB: The request of *by document* or *by resource* applies to the entire list of request IDs.
NB: The mechanism of matching a supplied request ID to a resource locator is not specified.
NB: The process currently does not handle attachments.

ToDo: Extend to produce (log) a usage record of the obtain.

API

```
POST <nodeURL>/obtain

Arguments:
        None

Request Object:                                   // list of resource data descriptions to obtain
        {
         "by_doc_ID":               boolean,      // request is for specific document for each ID
                                                  // request_ID is a doc_ID
                                                  // optional, default FALSE,

        "by_resource_ID":           boolean,      // request is for a collation of documents
                                                  // for each ID
                                                  // optional, default TRUE
```

```
                                                // request_ID is a resource_locator
         "ids_only":            boolean,        // request is just for IDs, not documents
                                                // optional, default FALSE
        "request_IDs":
        [                                       // array of
         "request_ID":   ID             // resource ID or
                                                // resource data description document ID
                                                // optional
                                                // if missing return documents for ALL IDs
        ]}

Results Object:                                 // list of resource data description documents
        {"documents":
         [
         {
          "doc_ID":      ID,                    // document ID
           [{resource_data_description}] // resource data description documents
                                                // array
                                                // present only if ID is valid, otherwise NULL
          }
         ]
        }

Return Codes:
        200
        500
```

Basic Obtain

```
// Obtain the resource data description documents for each supplied ID
IF by_doc_ID AND by_resource_ID
        THEN
                error // only one can be true
                EXIT
IF by_doc_ID
        IF request_ID not specified
                ▲THEN request_ID array := ALL doc_IDs
                ▲THEN set doc_IDs in request_ID array
                        //  based on the values in the service description (none, ALL, subset)
        FOR EACH request_ID
                Put the request_ID in the results object
                IF ids_only THEN SKIP
                GET the corresponding resource data description document
                IF Successful
                        THEN PUT the resource data description document in the results object
                                ▲//  all stored key-value pairs or only those defined in the spec
                                // as defined in the service description
                        ELSE PUT NULL in the results object
```

```
IF by_resource_ID
    IF request_ID not specified
        ▲THEN request_ID array := ALL unique resource_locators
        ▲THEN set unique_resource_locations in  request_ID array
            //  based on the values in the service description (none, ALL, subset)
    FOR EACH request_ID
        IF NOT ids_only
            THEN FIND the collation of resource data description documents
            WHERE resource_locator MATCHES supplied request_ID
        IF Successful
            PUT the request ID in the results object
            IF ids_only THEN SKIP
            FOR EACH resource data description document
                GET the corresponding resource data description document
                PUT the resource data description document in the results object
                    ▲// all stored key-value pairs or only those defined in the spec
                    // as defined in the service description
        ELSE PUT NULL in the results object
```

Service Description

```
{
 "doc_type":            “service_description”,
▲ "doc_version":                  “0.20.0”,
 "doc_scope":           “node”,
 "active":              true,
 "service_id":          “<uniqueid>”,
 "service_type":        “access”,
 "service_name":        “Basic Obtain”,
 "service_version":     “0.10.0”,
 "service_endpoint":    “<nodeURL>”,
▲”service_auth”:                      // service authentication and authorization descriptions
 {
▲"service_authz":       [“xxxx”],            // authz values for the service
▲“service_key”:                 T/F,         // does service use an access key

▲“service_https”:       T/F                  // does service require https
 }
▲ "service_data":
{
   “id_limit”:          integer,      // specify the maximum number off IDs
                                      // the service will return when requesting ALL
                                      // 0 means ALL is not a valid request
                                      // optional, return ALL if missing
   “doc_limit”:         integer,      // specify the maximum number of documents
```

```
                                        //  the service will return when requesting ALL
                                        // 0 means ALL is not a valid request
                                        // optional, return ALL if missing
   “spec_kv_only”:        boolean       // T to return only spec-defined key-value pairs
                                        // F to return all stored key-value pairs
                                        // optional, default F

  }

 }
```

Basic Harvest Service

The basic harvest service can be used by an external agent to connect to a node to harvest (pull) the resource data description documents held by the node. The service is patterned after the OAI-PMH specification. The service is designed to be extended to support full OAI-PMH–compliant harvesting.

The service can harvest the native JSON encoded metadata or paradata resource data, i.e., it harvests the resource data in the native format, not XML-encoded Dublin Core metadata or some other metadata dissemination. Harvest is done by resource data description document ID or by resource ID, i.e., by resource locator. Set-based harvesting is not currently supported. Flow control is not currently supported. OAI-PMH verbs are included directly in the HTTP path (rather than as an argument to provide a more RESTful API). Both GET and POST encoding of requests are supported.

▲The internal JSON storage of a resource data description document MAY include additional key-value pairs not defined in this specification. The service MAY return all stored key-value pairs, or only the key-value pairs defined in this specification for the resource data description document. The service document SHALL indicate if the returned documents are limited to the specification-defined key-value pairs only or if all stored key-value pairs are returned.

OAI-PMH Extension: IDs MAY be:

- the document ID for a resource data description document. The service SHALL return the single resource data description document that matches the ID.
- a unique resource identifier, e.g., the resource locator. The service SHALL return all resource data description documents for the specified resource that satisfy other harvest criteria.

Mapping of Learning Registry Basic Harvest to OAI-PMH Concepts

Native OAI-PMH Concept	**Learning Regisry Harvest API Concept**
Repository (harvest API end point)	Node Resource Data Description Document Database
Resource (something that has records)	Resource
Item (something in the repository for which a record can be disseminated)	Resource Data, e.g., an individual Resource Data Description Document or a collation of Resource Data Description Documents for a unique Resource

Record (dissemination output)	Resource Data Description Document, JSON Encoded
Item Identifier (URI)	Resource Data Description Document ID or Resouce ID/Resource Locator
Metadata Format	Resource Data Description Document JSON Object Schema
Set	*Sets for organizing resource data are not defined in this version of the specification*
GetRecord Verb	<nodelURL>harvest/getrecord
ListRecords Verb	<nodelURL>harvest/listrecords
ListIdentifiers Verb	<nodelURL>harvest/listidentifiers
Identify Verb	<nodelURL>harvest/identify
ListMetadataFormats Verb	<nodelURL>harvest/listmetadataformats
ListSets Verb	<nodelURL>harvest/listsets

Each of the six harvest verbs are specified separately. The Service Description document SHALL apply to the entire API.

The network node SHALL maintain a value for the earliest publication time for documents harvestable from the node (earliestDatestamp). Time-based harvesting MAY request harvest for documents published, updated or deleted after that time. The node MAY maintain documents with an earlier timestamp, but these documents SHALL NOT be accessible via harvest. The granularity for access via the timestamp MAY be days or seconds. The granularity of the timestamp SHALL be stored in the service description document.
NB: The actual timestamp MAY have a finer granularity.
NB: All times are UTC 0.

Get Record

The Get Record verb returns the resource data description documents for the specified resource data document ID or resource ID. If the request ID is a resource data description document ID, the service SHALL return the single resource data description document that matches the ID. If the request ID is a unique resource identifier, e.g., the resource locator, the service SHALL return all resource data description documents for the specified resource. The API only returns JSON. Different metadata formats cannot be specified. The service SHALL return complete resource data description documents.

NB: The process currently does not handle attachments.
NB: The default is that IDs are for resources, not documents.

ToDo: Extend to produce (log) a usage record of the harvest.

API

```
GET <nodeURL>/harvest/getrecord?request_id=<id>
        &by_doc_ID=<T|F>
        &by_resource_ID=<T|F>
POST <nodeURL>/harvest/getrecord

Arguments (HTTP GET):
        "request_ID":          ID,              // resource data description document ID or
                                                // resource ID
                                                // required
        "by_doc_ID":           boolean,         // request is for a single document
                                                // optional, default FALSE
                                                // request_ID is a doc_ID
        "by_resource_ID":      boolean          // request is for a collation of all documents
                                                // for the specified resource
                                                // optional, default TRUE
                                                // request_ID is a resource_locator

Arguments (HTTP POST):
        None

Request Object (HTTP GET):
        None

Request Object (HTTP POST):
        {"request_ID":         ID,              // resource data description document ID or
                                                // resource ID
                                                // required
         "by_doc_ID":          boolean,         // request is for a single document
                                                // optional, default FALSE
                                                // request_ID is a doc_ID
         "by_resource_ID",     boolean          // request is for a collation of all documents
                                                // for the specified resource
                                                // optional, default TRUE
                                                // request_ID is a resource_locator
        }

Results Object:
        {
         "OK":                 boolean,         // T if successful
         "error":              "string",        // text describing error
                                                // present only if NOT OK
         "responseDate":       "string",        // time of report, time/date encoding
         "request":                             // the API request
          {
```

```
   “verb”:                             “getrecord”,       // the literal “getrecord”
   “identifier”:              ID,                 // request  ID
   “by_doc_ID”:               boolean,            // request is for a single document
   “by_resource_ID”:          boolean,            // request is for a collation of documents
   “HTTP_request”:            “string”            // the HTTP request as a string
  },
 “getrecord”:                                     // the resource data description documents
                                                  // present only if ID is valid, otherwise NULL
  {
   “record”:                                      // record container
     [{
     “header”:                                    // header info
      {
       “identifier”:          ID,                 // resource data description document ID
       “datestamp”:           “string”,           // resource data timestamp, date/time
                                                  // requried, granularity of 1 second
       “status”:              “string”            // fixed vocabulary [“active”, “deleted”]
                                                  // optional, “active” if not present
      },
     “resource_data”:
      {resource_data_description}                 // resource data description documents
     }]
  }
}
```

Basic Harvest: GetRecord

```
    // Return the resource data description documents for the supplied ID
    Build results object
        responseDate := time of report              // time/date encoding
        request :=                                  // the API request
         {“verb”:                  “getrecord”,     // the literal “getrecord”
          “identifier”:            ID,              // request ID
          “by_doc_ID”:             boolean,         // request value
          “by_resource_ID””        boolean,         // request value
          “HTTP_request”:          “string”         // the HTTP request as a string
          }
        IF request_ID not supplied  // return error
            THEN   OK := FALSE
                    error := ‘badArgument”
                    EXIT
        IF by_doc_ID AND by_resource_ID
            THEN   OK := FALSE
                   error := “badArgument”  // only one can be true
                   EXIT
        IF by_resource_ID // get the list of documents otherwise it’s just the requested ID
            THEN   FIND the collation of resource data description document IDs
                   WHERE resource_locator MATCHES request <identifier>
        FOR EACH resource data description document ID
        GET the corresponding resource data description document
        IF successful
            THEN // return resource data
                   // header
                   datestamp := node_timestamp from the resource data description
                   identifier := resource data description document ID
                   IF delete_data_policy <> “no”
                       AND the resource data description document has been deleted
                       THEN status := “deleted”
                   // resource data
                   PUT the resource data description document in the results object
                       ▲//  all stored  key-value pairs or only those defined in the spec
                       // as defined in the service description
                   OK := TRUE
            ELSE // not found error
                   PUT NULL in the results object
                   OK := FALSE
                   error := “idDoesNotExist”
    TRANSFORM results to specified CONTENT-TYPE
```

List Records

The List Records verb returns the resource data description documents for document added to the node within a specified time/date range. The API only returns JSON. The service SHALL return complete resource data description documents. Different metadata formats cannot be specified. Flow control is not currently supported. Set-based harvesting is not currently supported. Return of attachments is not currently supported.

▲*NB*: List records does not support access by resource locator. Documents may only be accessed by document ID.
ToDo: Extend to produce (log) a usage record of the harvest.

API

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

    Arguments (HTTP GET):
        “from”:         “string”,           // start of harvest time/date range
                                            // optional, time/date
                                            // earliest resource data timestamp if not present
        “until”:        “string”            // end of harvest time/date range
                                            // optional, time/date
                                            // latest resource data timestamp if not present

    Arguments (HTTP POST):
        None
```

Request Object (HTTP GET):

```
None
```

Request Object (HTTP POST):

```
{
 “from”:          “string”,              // start of harvest time/date range
                                         // optional, time/date
                                         // earliest resource data timestamp if not present
 “until”:         “string”               // end of harvest time/date range
                                         // optional, time/date
                                         // latest resource data timestamp if not present
}
```

Results Object:

```
{
 "OK":                  boolean,          // T if successful
 "error":               “string”,         // text describing error
                                          // present only if NOT OK
 “responseDate”:        “string”,         // time of report, time/date encoding
 “request”:                               // the API request
 {
  “verb”:               “listrecords”,    // the literal “listrecords”
  “from”:               “string”,         // specified start of harvest time/date range
                                          // time/date
  “until”:              “string”.         // specified end of harvest time/date range
                                          // time/date
  “HTTP_request”:       “string”          // the HTTP request as a string
  },
 “listrecords”:                           // array of records
  [
   “record”:                              // the resource data description document
                                          // present only if ID is valid, otherwise NULL
   {
    “header”:
    {
     “identifier”:      ID,               // resource data description document ID
     “datestamp”:       “string”,         // resource data timestamp, date/time
                                          // required, granularity of 1 second
     “status”:          “string”,         // fixed vocabulary [“active”, “deleted”]
                                          // optional, “active” if not present
    }
    resource_data:
    {resource_data_description}           // resource data description documents
    }
   ]
}
```

Basic Harvest: ListRecords

```
// Return the resource data description documents for the specified time range
Build results object
    responseDate := time of report              // time/date encoding
    request :=                                  // the API request
    {"verb":                  "listrecords",    // the literal "listrecords"
     "from":                  "string",         // specified start of harvest time/date range
     "until":                 "string",         // specified end of harvest time/date range
     "HTTP_request":          "string"          // the HTTP request as a string
     }
    IF from > until  // return error
        THEN   OK := FALSE
               error := 'badArgument"
               EXIT
    IF granularity of from time <> granularity of until time // return error
        THEN   OK := FALSE
               error := 'badArgument"
               EXIT
    IF granularity of from time < service granularity
        // request is for seconds, service instance only supports days (not seconds)
        THEN   OK := FALSE
               error := 'badArgument"
               EXIT
    IF from not specified THEN from := earliest timestamp
    IF until not specified THEN until := latest timestamp
    FOR EACH resource data description document
        IF from <=  node_timestamp from the resource data description document
            <= until // timestamp inclusive in [from:until] range
        THEN
            // return header for resource data
            datestamp := node_timestamp from the resource data description
            identifier := resource data description document ID
            IF the delete_data_policy <> "no"
                AND the resource data description document has been deleted
                THEN status := "deleted"
            // return the resource data
            PUT the resource data description document in the results object
    IF listrecords array is empty
        THEN
            OK := FALSE
            error := "noRecordsMatch"
        ELSE
            OK := TRUE
TRANSFORM results to specified CONTENT-TYPE
```

List Identifiers

The List Identifiers verb returns the OAI-PMH header information from the resource data description documents for the specified resource data document IDs within a specified time/date range. The API only returns JSON. Different metadata formats cannot be specified. Flow control is not currently supported. Set-based harvesting is not currently supported.

The API is functionally equivalent to the List Records API, only header information returned; no resource data is returned. Data elements are renamed to map to the the OAI-PMH specification.

NB: There is currently no mechanism to return the collection of ids of resources where a new resource data description document has been added to the collation of documents for a resource within the specified time range. ▲ Documents may only be accessed by document ID.

API

```
GET <nodeURL>/harvest/listidentifiers?from=<date>&until=<date>
POST <nodeURL>/harvest/listidentifiers

Arguments (HTTP GET):
        “from”:         “string”,               // start of harvest time/date range
                                                // optional, time/date
                                                // earliest resource data timestamp if not present
        “until”:        “string”                // end of harvest time/date range
                                                // optional, time/date
                                                // latest resource data timestamp if not present

Arguments (HTTP POST):
        None

Request Object (HTTP GET):
        None

Request Object (HTTP POST):
        {
         “from”:        “string”,               // start of harvest time/date range
                                                // optional, time/date
                                                // earliest resource data timestamp if not present
         “until”:       “string”                // end of harvest time/date range
                                                // optional, time/date
                                                // latest resource data timestamp if not present
        }

Results Object:
        {
         "OK":                  boolean,        // T if successful
         "error":               “string”,       // text describing error
                                                // present only if NOT OK
         “responseDate”:        “string”,       // time of report, time/date encoding
“request”:                                    // the API request
{
 “verb”:                “listidentifiers”,   // the literal “listidentifiers”
 “from”:                “string”,            // specified start of harvest time/date range
                                             // time/date
 “until”:               “string”.            // specified end of harvest time/date range
                                             // time/date
 “HTTP_request”:        “string”             // the HTTP request as a string
 },
“listidentifiers”:                           // array of headers
 [
  “header”:
  {
   “identifier”:        ID,                  // resource data description document ID
   “datestamp”:         “string”,            // resource data timestamp, date/time
                                             // requried, granularity of 1 second
   “status”:            “string”,            // fixed vocabulary [“active”, “deleted”]
                                             // optional, “active” if not present

  }
 ]
}
```

Basic Harvest: ListIdentifiers

```
    // Return the resource data description document headers for the specified time range
    Build results object
        responseDate := time of report       // time/date encoding
        request :=                           // the API request
        {“verb”:            “listidentifiers”,   // the literal “listidentifiers”
         “from”:            “string”,            // specified start of harvest time/date range
         “until”:           “string”,            // specified end of harvest time/date range
         “HTTP_request”:    “string”             // the HTTP request as a string
         }
        IF from > until  // return error
                THEN   OK := FALSE
                        error := ‘badArgument”
                        EXIT
        IF granularity of from time <> granularity of until time // return error
                THEN   OK := FALSE
                        error := ‘badArgument”
                        EXIT
        IF  granularity of from time < service granularity
                // request is for seconds, service instance only supports days (not seconds)
                THEN   OK := FALSE
                        error := ‘badArgument”
                        EXIT
        IF from not specified THEN from := earliest timestamp
        IF until not specified THEN unti := latest timestamp
        FOR EACH resource data description document
            IF from <= node_ timestamp from the resource data description document
                <= until // timestamp inclusive in [from:until] range
            THEN
                // return header for resource data
                datestamp := node_timestamp from the resource data description
                identifier := resource data description document ID
                IF the delete_data_policy <> “no”
                    AND the resource data description document has been deleted
                    THEN status := “deleted”
        IF listidentifiers array is empty
            THEN
                OK := FALSE
                error := “noRecordsMatch”
            ELSE
                OK := TRUE
    TRANSFORM results to specified CONTENT-TYPE
```

Identify

The Identify verb returns a description of the harvest end point. The service SHALL return the values in JSON. The service SHALL return all of the key-value pairs listed. The service MAY return additional key-value pairs that describe the harvest service.

A network node SHALL maintain all of the data necessary to return the required key-value pairs.

API

```
    GET <nodeURL>/harvest/identify
    POST <nodeURL>/harvest/identify

    Arguments:
        None

    Request Object:
        None

    Results Object:
        {
         "OK":                  boolean,        // T if successful
         "error":               “string”,       // text describing error
                                                // present only if NOT OK
         “responseDate”:        “string”,       // time of report, time/date encoding
         “request”:                             // the API request
         {
          “verb”:               “identify”,     // the literal “identify”
          “HTTP_request”:       “string”         // the HTTP request as a string
          },
         “identify”:
```

```
{
 "node_id":              "string",         // ID of the network node
 "repositoryName":       "string",         // name of the network node
 "baseURL":              "string",         // URL of the network node
 "protocolVersion":      "2.0",            // the literal "2.0"
 "service_version":      "string",         // version of th Harvest service API
 "earliestDatestamp":    "string",         // time/date encoding
 "deletedRecord":        "string",         // node delete policy
 "granularity",          "string",         // granularity from the service policy
 "adminEmail",           "string"          // node admin URL
 }
}
```

Basic Harvest: Identify

```
    // Return the description of the harvest service
    Build results object
        OK := TRUE
        responseDate := time of report // time/date encoding
        request := // the API request
         {"verb":                "identify",       // the literal "identify"
          "HTTP_request":        "string"           // the HTTP request as a string
          }
        node_id := node_id from the network node description
        repositoryName := node_name from the network node description
        baseURL := <nodeURL> // URL of the network node
        protocolVersion := "2.0" // the OAI-PMH version
        service_version := service_version from the Harvest service description
        earliestDatestamp := timestamp
            // the oldest guaranteed publish/update or delete timestamp
            // time/date encoding with service-specified granularity
        deletedRecord := deleted_data_policy from the node_policy from the
             network node description
        granularity := granularity from the Harvest service description
        ▲adminEmail := node_admin_identity from the network node description
    TRANSFORM results to specified CONTENT-TYPE
```

List Metadata Formats

The List Metadata Formats verb returns the list of metadata formats available for harvests. The harvest API only returns JSON encoded resource data descriptions: this is the only metadata format defined in the service description. The metadataPrefix SHALL be the value specified in the metadataformats structure in the service description (e.g., "LR_JSON_0.10.0"). The service SHALL return all of the key-value pairs listed. The service SHALL NOT return additional key-value pairs.

The services does not support the retrieval of the metadata format for an individual resource data description document. Including a ID in the request SHOULD produce an error.

API

```
GET <nodeURL>/harvest/listmetadataformats
POST <nodeURL>/harvest/listmetadataformats

Arguments:
        None

Request Object:
        None

Results Object:
        {
         "OK":                    boolean,          // T if successful
         "error":                 “string”,         // text describing error
                                                    // present only if NOT OK
         “responseDate”:          “string”,         // time of report, time/date encoding
         “request”:                                 // the API request
         {
          “verb”:                 “listmetadataformats”,   // the literal “listmetadataformats”
          “HTTP_request”:         “string”           // the HTTP request as a string
          },
         “listmetadatadormats”:                     // array of supported metadata formats
          [
           “metadataformat”:
           {
            “metadataPrefix”: “string”              // metadata format name/prefix
                                                    // other elements will go here
           }
          ]
        }
```

Basic Harvest: List Metadata Formats

```
    // Return the description of the metadata formats supported for harvest
    Build results object
        OK := TRUE
        responseDate := time of report // time/date encoding
        request := // the API request
         {“verb”:                 “listmetadataformats”,   // the literal “listmetadataformats”
          “HTTP_request”:         “string”                 // the HTTP request as a string
          }
        metadataFormat := metadataformat structure from the Harvest service description
            // the key-value pair [{“metadataPrefix”: “LR_JSON_0.10.0”}]
    TRANSFORM results to specified CONTENT-TYPE
```

List Sets

The List Sets verb returns the list of sets used to organize resource data descriptions. Support for sets is not defined in this version of the specification. The API SHALL return a standard error indicating that sets are not available.

API

```
GET <nodeURL>/harvest/listsets
POST <nodeURL>/harvest/listsets

Arguments:
        None

Request Object:
        None

Results Object:
        {
         "OK":            boolean,                 // T if successful
         "error":         "string",                // text describing error
                                                   // present only if NOT OK
         "responseDate":          "string",        // time of report, time/date encoding
         "request":                                // the API request
         {
          "verb":                 "listsets",      // the literal "listsets"
          "HTTP_request":         "string"          // the HTTP request as a string
          }
        }
```

Basic Harvest: List Sets

```
// Return the description of the sets available for harvest
Build results object
        OK := FALSE
        error := "noSetHierarchy"
        responseDate := time of report // time/date encoding
        request := // the API request
         {"verb":                 "listsets",      // the literal "listsets"
          "HTTP_request":         "string"          // the HTTP request as a string
          }
TRANSFORM results to specified CONTENT-TYPE
```

Service Description

```
{
 "doc_type":               "service_description",
 ▲"doc_version":                   "0.20.0",
 "doc_scope":              "node",
 "active":                 true,
 "service_id":             "<uniqueid>",
 "service_type":           "access",
 "service_name":           "Basic Harvest",
 "service_version":     “0.10.0”,
 "service_endpoint":    “<nodeURL>”,
▲”service_auth”:                         // service authentication and authorization descriptions
 {
▲"service_authz":       [“xxxx”],                // authz values for the service
▲“service_key”:         T/F,             // does service use an access key
▲“service_https”:       T/F              // does service require https
 }
 “service_data”:
 {
  “granularity”:        “string”,                // literal fixed vocabulary
                                         // “YYYY-MM-DD” (day granularity)
                                         // or “YYYY-MM-DDThh:mm:ssZ” (second granularity)
  “flow_control”:       FALSE,           // flow control not supported
  “setSpec”:            NULL,            // sets are not supported
 ▲ “spec_kv_only”:      boolean.         // T to return only spec-defined key-value pairs
                                         // F to return all stored key-value pairs
                                         // optional, default F
  “metadataformats”:                     // array of supported metadata formats
  [
   “metadataFormat”:                     // description of a metadata format
   {
    “metadataPrefix”:   “LR_JSON_0.10.0”         // the only supported harvest form
                                                 // the Full OAI-PMH service will define
                                                 // schema and metadataNamespace
                                                 // where appropriate
   }
  ]
 }
}
```

OAI-PMH Harvest Service

The OAI-PMH harvest services can be used by an external agent to connect to a node to harvest (pull) the resource data (e.g., the metadata or paradata) contained in the resource data description documents stored at the node. The service defines how to harvest a variety of metadata formats (DC, LOM), paradata formats, etc., along with full resource data description documents stored at the node. Unless specified, the service SHALL support OAI-PMH V2.0. Harvest is done by resource data description document ID or by resource ID, i.e., by resource locator. Set-based harvesting is not currently supported. Flow control is not currently supported.

OAI-PMH Extension: IDs MAY be:

- the document ID for a resource data description document. The service SHALL return the single resource data description document that matches the ID.
- a unique resource identifier, e.g., the resource locator. The service SHALL return all resource data description documents for the specified resource that satisfy other harvest criteria.

NB: The service could be built using basic harvest service. The core functionality is present in basic harvest service. A transformation would be applied to the results to convert them from JSON to XML.

To support extensions, the OAI-PMH XSD has been extended. A copy of the schema is currently available at: http://www.learningregistry.org/documents/downloads/OAI-PMH-LR.xsd
This schema:

- adds the ID arguments for GetRecord
- supports the return of a multiple records from GetRecord
- adds the ID arguments for ListMetatdataFormats
- makes metadataNamespace optional

NB: There is no guarantee of persistence of the XSD. The service description for the OAI-PMH harvest service includes a schema location key-value pair used to indicate the persistent XSD location.

Mapping Learning Registry OAI-PMH Harvest to OAI-PMH Concepts

Native OAI-PMH Concept	**Learning Registry Harvest API Concept**
Repository (harvest API end point)	Node Resource Data Description Document Database
Resource (something that has records)	Resource
Item (something in the repository for which a record can be disseminated)	Resource Data, e.g., an individual Resource Data Description Document or a collation of Resource Data Description Documents for a unique Resource
Record (dissemination output)	Resource Data Description Document Resource Data
Item Identifier (URI)	Resource Data Description Document ID or Resource ID/Resource Locator
Metadata Format	Resource Data Description Document Payload Schema
Set	*Sets for organizing resource data are not defined in this version of the specification*
GetRecord Verb	<nodelURL>OAI-PMH?verb=GetRecord
ListRecords Verb	<nodelURL>OAI-PMH?verb=ListRecords
ListIdentifiers Verb	<nodelURL>OAI-PMH?verb=ListIdentifiers
Identify Verb	<nodelURL>OAI-PMH?verb=Identify
ListMetadataFormats Verb	<nodelURL>OAI-PMH?verb=ListMetadataFormats

ListSets Verb	<nodelURL>OAI-PMH?verb=ListSets

Each of the six harvest verbs are specified separately. The Service Description document SHALL apply to the entire API.

The network node SHALL maintain a value for the earliest publication time for documents harvestable from the node (earliestDatestamp). Time-based harvesting MAY request harvest for documents published, updated or deleted after that time. The node MAY maintain documents with an earlier timestamp, but these documents SHALL NOT be accessible via harvest. The granularity for access via the timestamp MAY be days or seconds. The granularity of the timestamp SHALL be stored in the service description document.
NB: The actual timestamp MAY have a finer granularity.
NB: All times are UTC 0.

OAI-PMH Extension: If the requested dissemination format in metadataPrefix matches the JSON metadataPrefix in the servcie description (e.g., “LR_JSON_0.10.0”), the service SHALL behave as the basic harvest service, i.e., it returns the complete resource data description document as JSON.

▲The internal JSON storage of a resource data description document MAY include additional key-value pairs defined in this specification. The service MAY return all stored key-value pairs, or only the key-value pairs defined in this specification for the resource data description document. The service document SHALL indicate if the returned documents are limited to the specification-defined key-value pairs only or if all stored key-value pairs are returned.

Get Record

The Get Record verb returns resource data (e.g., the metadata or paradata) that matches the requested dissemination format for the specified resource data description document ID or resource ID.

OAI-PMH Extension: If the request ID is a resource data description document ID, the service SHALL return the metadata dissemination for the single resource data description document that matches the ID. If the request ID is a unique resource identifier, e.g., the resource locator, the service SHALL return the metadata disseminations for all resource data description documents for the specified resource.

The Get Record verb SHALL support the return any resource_data that matches the requested dissemination format that is associated with the requested resource data document, i.e., any payload where the payload_schema matches the requested dissemination format. An implementation MAY support the translation of the stored resource_data to the requested dissemination format. An implementation MAY support equivalence matching for the requested dissemination format, e.g., the available format X is recognized to be the same as the requested format Y. An implementation MAY support the automated generation of resource_data in the requested dissemination format.

The Get Record verb SHALL support the return of resource_data independent of where it is stored in the payload, i.e., it returns any inline, attached or referenced resource data in the payload of the specified resource data description document.

If the requested metadata dissemination is not available for the requested ID, the service SHALL return a cannotDisseminateFormat error.

OAI-PMH Extension: If the requested dissemination format in metadataPrefix matches the JSON metadataPrefix in the servcie description (e.g., “LR_JSON_0.10.0”), the service SHALL behave as the basic harvest service, i.e., it returns the complete resource data description document as JSON. This behavior is NOT specified in the pseudo code below.

ToDo: Extend to produce (log) a usage record of the harvest.

API

```
    GET<nodeURL>/OAI-PMH?verb=GetRecord
        &identifier=<ID>
        &metadataPrefix=<resourcedataformat>
        &by_doc_ID=<T|F>
        &by_resource_ID=<T|F>

    POST <nodeURL>/OAI-PMH
    Post Payload: verb=GetRecord
        &identifier=<ID>
        &metadataPrefix=<resourcedataformat>
        &by_doc_ID=<T|F>
        &by_resource_ID=<T|F>

    Request Key-Value Pairs (as per OAI-PMH Specification, with Learning Registry extensions)
        verb              = GetRecord          // literal “GetRecord”, required
        identifier        = <string>           // resource data description document ID
                                               // required
        metadataPrefix    = <string>           // requested metadata dissemination format
                                               // required
        by_doc_ID         = boolean            // request is for a single document
                                               // optional, default FALSE
                                               // identifier is a doc_ID
                                               // OAI-PMH extension
        by_resource_ID    = boolean      ,     // request is for a collation of all documents
                                               // for the specified resource
                                               // optional, default TRUE
                                               // identifier is a resource_locator
                                               // OAI-PMH extension

    Results XML
        Well formed XML instance document that validates according to the Learning Registry
        extended OAI-PMH XML XSD
        Contains:
            <responseDate />            // required XML element
            <request />                 // required XML element
                                        // includes extensions
            <error />                   // XML element if errors
            <GetRecord />               // XML element with results if no errors
```

OAI-PMH: GetRecord

```
    // Return the resource data from the resource data description document for the ID in the request
    Build XML results document
    EMIT OAI-PMH namespace declarations
    EMIT the required + extension elements
        <responseDate>time of report<responseDate>
        <request
            verb="GetRecord"                        // the literal "GetRecord"
            identifier=<ID>                         // request ID
            metadataPrefix=<metadataformat>         // requested metadata format
            by_doc_ID=<boolean>                     // by document request flag
            by_resource_ID=<boolean>                // by resource request flag
            >
            HTTP_request                            // the HTTP request as a string
        </request>
    IF identifier not supplied  // return error element
        <error code="badArgument" />
        Complete XML
        EXIT
    IF metadataPrefix not supplied // return error element
        <error code="badArgument" />
        Complete XML
        EXIT
    IF by_doc_ID AND by_resource_ID
        <error code="badArgument" /> // only one can be true
        Complete XML
        EXIT

    // Does the document exist
    IF by_doc_ID AND
        no resource data description document with doc_ID  = <identifier>
        THEN   <error code="idDoesNotExist" />
               Complete XML
               EXIT
    IF by_resource_ID AND no resource data description document with resource_locator  =
    <identifier>
        THEN   <error code="idDoesNotExist" />
               Complete XML
               EXIT
    IF by_resource_ID // get the list of documents otherwise it's just the requested ID
        THEN   FIND the collation of resource data description documents IDs as <identifier>
```

```
                WHERE resource_locator MATCHES request <identifier>

    FOR EACH resource data description document IDs
    // Is there an acceptable metadata format
    IF payload_schema <> <resourcedataformat> OR
            NOT Same_As or Translatable (payload_schema, <resourcedataformat>)
            <error code="cannotDisseminateFormat" />
            Complete XML
            EXIT
    Build <GetRecord>
    <GetRecord>
    Build <record>
    <record>
            EMIT <header>
            <header
                    IF delete_data_policy <> "no"
                    AND the resource data description document has been deleted
                    THEN status ="deleted"
                    >
            <identifier>resource data description document doc_ID</identifier>
            <datastamp>node_timestamp from the resource data description</datestamp>
            </header>
            EMIT <metadata>
            <metadata>
                    CASE
                            payload_placement = "inline"
                                    EMIT resource data in XML
                            payload_placement = "attachment"
                                    EMIT attached document in XML
                            payload_placement = "linked"
                                    Get resource data from payload_schema_locator
                                    EMIT document in XML
                    IF EMIT fails
                            <error code="cannotDisseminateFormat" />
                            Complete XML
                            EXIT
            </metadata>
    </record>
    </GetRecord>
```

List Records

The List Records verb returns the resource data description documents for the specified resource data document IDs within a specified time/date range. Set-based harvesting is not currently supported.

The List Records verb SHALL support the return of any resource_data that matches the requested dissemination format that is associated with the specified resource data document, i.e., any payload where the payload_schema matches the requested dissemination format. An implementation MAY support the translation of the stored resource_data to the requested dissemination format. An implementation MAY support equivalence matching for the requested dissemination format, e.g., the available format X is recognized to be the same as the requested format Y. An implementation MAY support the automated generation of resource_data in the requested dissemination format.

The List Records verb SHALL support the return of resource_data independent of where it is stored in the payload, i.e., it returns any inline, attached or referenced resource data in the payload of the specified resource data description document.

NB: The combination of processing deleted records and records that do not have the specified metadata dissemination is not clear in the OAI-PMH specification. Since not all resource data description documents support all formats, the service only returns deleted status for documents that match the requested dissemination format.

NB: A test to determine if no records match the requested metadata dissemination format is not included. The resulting error code of cannotDisseminateFormat does not occur. If no records match the requested metadata dissemination format, the error code SHALL be noRecordsMatch.

OAI-PMH Extension: If the requested dissemination format in metadataPrefix matches the JSON metadataPrefix in the servcie description (e.g., “LR_JSON_0.10.0”), the service SHALL behave as the basic harvest service, i.e., it returns the complete resource data description document as JSON. This behavior is NOT specified in the pseudo code below.

▲*NB*: List records does not support access by resource locator. Documents may only be accessed by document ID.

ToDo: Extend to produce (log) a usage record of the harvest.

API

```
GET<nodeURL>/OAI-PMH?verb=ListRecords
        &from=<date>
        &until=<date>
        &metadataPrefix=<resourcedataformat>

POST <nodeURL>/OAI-PMH
Post Payload: verb=ListRecords
        &from=<date>
        &until=<date>
        &metadataPrefix=<resourcedataformat>

Request Key-Value Pairs (as per OAI-PMH Specification, with Learning Registry extensions)
        verb            = ListRecords           // literal “ListRecords”, required
        from            =<date>                 // start of harvest time/date range
                                                // optional, time/date
                                                // earliest resource data timestamp if not present
        until           =<date>                 // end of harvest time/date range
                                                // optional, time/date
                                                // latest resource data timestamp if not present
        metadataPrefix  = <string>              // requested metadata dissemination format
                                                // required

    Results XML
        Well formed XML instance document that validates according to the Learning Registry
        extended OAI-PMH XML XSD
        Contains:
            <responseDate />                    // required XML element
            <request />                         // required XML element
            <error />                           // XML element if errors
            <ListRecords />                     // XML element with results if no errors
```

OAI-PMH: ListRecords

```
    // Return the resource data description documents for the specified time range
    Build XML results document
    EMIT OAI-PMH namespace declarations
    EMIT the required elements
        <responseDate>time of report<responseDate>
        <request
            verb="ListRecords"                      // the literal "ListRecords"
            metadataPrefix=<metadataformat>         // requested metadata format
            from=<date>                             // start of harvest time/date range
            until=<date>                            // end of harvest time/date range
            >
            HTTP_request                            // the HTTP request as a string
        </request>
    IF from > until  // return error
        <error code="badArgument" />
        Complete XML
        EXIT
    IF granularity of from time <> granularity of until time // return error
        <error code="badArgument" />
        Complete XML
        EXIT
    IF granularity of from time < service granularity
        // request is for seconds, service instance only supports days (not seconds)
        <error code="badArgument" />
        Complete XML
        EXIT
    IF from not specified THEN from := earliest timestamp
    IF until not specified THEN until := latest timestamp
    Build <ListRecords>
    <ListRecords>
    FOR EACH resource data description document
        IF from <=  node_timestamp from the resource data description document
            <= until // timestamp inclusive in [from:until] range
        THEN
        IF payload_schema <> <resourcedataformat> OR
            NOT *Same_As or Translatable* (payload_schema, <resourcedataformat>)
            NEXT LOOP
        THEN
        Build a <record>
        <record>
        EMIT <header>
        <header
            IF delete_data_policy <> “no”
            AND the r*esource data description* document has been deleted
            THEN status =“deleted”
            >
        <identifier>resource data description document ID</identifier>
        <datastamp>node_timestamp from the r*esource data description*</datestamp>
        </header>
        EMIT <metadata>
        <metadata>
            CASE
                payload_placement = “inline”
                    EMIT resource data  in XML
                payload_placement = “attachment”
                    EMIT attached document in XML
                payload_placement = “linked”
                    Get resource data from payload_schema_locator
                    EMIT document in XML
            IF EMIT fails
                <error code=”cannotDisseminateFormat” />
                Complete XML
                EXIT
        </metadata>
    </record>
    </ListRecords>
    IF <ListRecords> is empty
        THEN
        DELETE <ListRecords> element
        <error code=”noRecordsMatch” />
        Complete XML
        EXIT
```

List Identifiers

The List Identifiers verb returns the header information for the resource data description documents for the specified resource data document IDs within a specified time/date range. Flow control is not currently supported. Set-based harvesting is not s currently supported.

The API is functionally equivalent to the List Records API, only header information is returned; no resource data is returned.

NB: There is currently no mechanism to return the collection of ids of resources where a new resource data description document has been added to the collation of documents for a resource within the specified time range. ▲ Documents may only be accessed by document ID.

API

```
    GET<nodeURL>/OAI-PMH?verb=ListIdentifiers
        &from=<date>
        &until=<date>
        &metadataPrefix=<resourcedataformat>

    POST <nodeURL>/OAI-PMH
    Post Payload: verb=ListIdentifiers
        &from=<date>
        &until=<date>
        &metadataPrefix=<resourcedataformat>

    Request Key-Value Pairs (as per OAI-PMH Specification)
        verb             = ListIdentifiers      // literal “ListIdentifiers”, required
        from             =<date>                // start of harvest time/date range
                                                // optional, time/date
                                                // earliest resource data timestamp if not present
        until            =<date>                // end of harvest time/date range
                                                // optional, time/date
                                                // latest resource data timestamp if not present
        metadataPrefix  = <string>              // requested metadata dissemination format
                                                // required

    Results XML
        Well formed XML instance document that validates according to the OAI-PMH XML XSD
        Contains:
            <responseDate />                    // required XML element
            <request />                         // required XML element
            <error />                           // XML element if errors
            <ListIdentifiers />                 // XML element with results if no errors
```

OAI-PMH: ListIdentifiers

```
    // Return the resource data description document headers for the specified time range
    Build XML results document
    EMIT OAI-PMH namespace declarations
    EMIT the required elements
        <responseDate>time of report<responseDate>
        <request
            verb=”ListIdentifiers”                  // the literal “ListIdentifiers”
            metadataPrefix=<metadataformat>         // requested metadata format
            from=<date>                             // start of harvest time/date range
            until=<date>                            // end of harvest time/date range
            >
            HTTP_request                    // the HTTP request as a string
        </request>
    IF from > until  // return error
        <error code=”badArgument” />
        Complete XML
        EXIT
    IF granularity of from time <> granularity of until time // return error
        <error code=”badArgument” />
        Complete XML
        EXIT
    IF granularity of from time < service granularity
        // request is for seconds, service instance only supports days (not seconds)
        <error code=”badArgument” />
        Complete XML
        EXIT
    IF from not specified THEN from := earliest timestamp
    IF until not specified THEN until := latest timestamp
    Build <ListIdentifiers>
    <ListListIdentifers>
    FOR EACH resource data description document
        IF from <=  node_timestamp from the resource data description document
            <= until // timestamp inclusive in [from:until] range
        THEN
        IF payload_schema <> <resourcedataformat> OR
            NOT Same_As or Translatable (payload_schema, <resourcedataformat>)
            NEXT LOOP
        THEN
        Build a <record>
        <record>
        EMIT <header>
        <header
            IF delete_data_policy <> “no”
            AND the resource data description document has been deleted
            THEN status =“deleted”
            >
        <identifier>resource data description document ID</identifier>
        <datastamp>node_timestamp from the resource data description</datestamp>
        </header>
    </record>
    </ListRecords>
    IF <ListRecords> is empty
        THEN
        DELETE <ListRecords> element
        <error code=”noRecordsMatch” />
        Complete XML
        EXIT
```

Identify

The Identify verb returns a description of the OAI-PMH harvest end point. The service SHALL return all of the values specified in the OAI-PMH specification, using the specified XML schema. The service MAY return additional XML elements that describe the harvest service specified in the OAI-PMH specification.

A network node SHALL maintain all of the data necessary to return the required elements.

API

```
        GET<nodeURL>/OAI-PMH?verb=Identify

        POST <nodeURL>/OAI-PMH
        Post Payload: verb=Identify

        Request Key-Value Pairs (as per OAI-PMH Specification)
                verb            = Identify              // literal “Identify”, required

        Results XML
                Well formed XML instance document that validates according to the OAI-PMH XML XSD
                Contains:
                        <responseDate />                // required XML element
                        <request />                     // required XML element
                        <error />                       // XML element if errors
                        <Identify/>                     // XML element with results if no errors
```

OAI-PMH: Identify

```
        // Return the description of the harvest service
        Build XML results document
        EMIT OAI-PMH namespace declarations
        EMIT the required elements
                <responseDate>time of report<responseDate>
                <request
                        verb=”Identify”                 // the literal “Identify”
                        >
                        HTTP_request                    // the HTTP request as a string
                </request>
        Build <Identify>
        EMIT the required elements
        <Identitfy>
                <repositoryName>node_name from the network node description</repositoryName>
                <baseURL>URL of the network node</baseURL>
                <protocolVersion>2.0</protocolVersion>
                <earliestDatestamp>the oldest guaranteed publish/update or delete
                        timestamp</earliestDatestamp>
                <deletedRecord>deleted_data_policy from the node_policy from the
                        network node description</deletedRecord>
                <granularity>granularity from the Harvest service description</granularity>
                ▲<adminEmail>node_admin_identity from the network node description</adminEmail>
        </Identify>
```

List Metadata Formats

The List Metadata Formats verb returns the list of metadata formats available for harvests. The service SHALL return all of the elements specified in the OAI-PMH specification, using the specified XML schema. The service SHALL NOT return additional XML elements.

The metadata format is a triple of three XML elements: <metadataPrefix>, <schema> and <metadataNameSpace>. The service determines the available formats from the payload_schema key-value pair in the resource data description documents. Each value in the payload_schema array SHALL be considered as a separate dissemination format, i.e., a separate value for <metadataPrefix>. The value for <schema> SHALL be the value of corresponding payload_schema_locator.

Determining the value of <metadataNameSpace> is optional. The service does not define how to determine the value for <metadataNameSpace>.

NB: Both <schema> and <metadataNameSpace> are optional elements in the <metadataFormat>.

If an identifier is provided, the metadata formats SHALL be returned only for the identified resource data description documents. If an identifier is *not* provided, the metadata formats SHALL be returned for *all* resource data description documents.

OAI-PMH Extension: If the request ID is a resource data description document ID, the service SHALL return the metadata formats for the single resource data description document that matches the ID. If the request ID is a unique resource identifier, e.g., the resource locator, the service SHALL return the metadata format for all resource data description documents for the specified resource.

Only unique dissemination formats SHALL be included in the list of formats. Duplicate dissemination formats SHALL be removed. A duplicate SHALL have identical <metadataPrefix>, <schema> and <metadataNameSpace> values to those of another entry. Two dissemination formats that differ in both <schema> or <metadataNameSpace> values SHALL be considered to be unique. Two dissemination formats that differ in only <schema> values SHALL be considered to be unique unless the service can determine that the actual schemata are identical copies. Determining if two schemata values represent identical copies is optional.

Values for payload_schema that correspond to generic schemata (e.g., “XML”, “RDF”) SHOULD be removed from the list of dissemination formats.

The service MAY order the resulting list of formats by the occurrences, most common first.

The service SHOULD NOT return values that do not satisfy the OAI-PMH requirement that

<metadataPrefix> be a string of "any valid URI unreserved characters".

The service SHALL include the Learning Registry JSON resource data description document format metadataPrefix specified in the metadataformats structure in the service description (e.g., "LR_JSON_0.10.0") in the results list of formats.

API

```
        GET<nodeURL>/OAI-PMH?verb=ListMetadataFormats
                &identifier=<id>
                &by_doc_ID=<T|F>
                &by_resource_ID=<T|F>

        POST <nodeURL>/OAI-PMH
        Post Payload: verb=ListMetadataFormats
                &identifier=<id>
                &by_doc_ID=<T|F>
                &by_resource_ID=<T|F>

        Request Key-Value Pairs (as per OAI-PMH Specification,with Learning Registry extensions)
                verb            = ListMetadataFormats  // literal "ListMetadataFormats", required
                identifier      = <string>             // resource data description document ID
                                                       // optional
                by_doc_ID       = boolean              // request is for a single document
                                                       // optional, default FALSE
                                                       // OAI-PMH extension
                by_resource_ID  = boolean       ,      // request is for a collation of all documents
                                                       // for the specified resource
                                                       // optional, default TRUE
                                                       // OAI-PMH extension

        Results XML
                Well formed XML instance document that validates according to the Learning Registry
                extended OAI-PMH XML XSD
                Contains:
                        <responseDate />               // required XML element
                        <request />                    // required XML element
                                                       // includes extensions
                        <error />                      // XML element if errors
                        <ListMetadataFormats />        // XML element with results if no errors
```

OAI-PMH: List Metadata Formats

```
        // Results View
        Define a view of the resource data description documents
                IF identifier is provided
                        THEN
                        IF by_doc_ID
                                THEN use the resource data description document where
```

```
                              doc_ID = <identifier>
                IF by_resource_ID
                        THEN use all resource data description documents where
                                resource_locator = <identifier>
                ELSE use all resource data description documents
        View includes: payload_schema, payload_schema_locator
        Expand to one payload_schema_locator for each value in payload_schema
        Optionally order by (1) payload_schema, (2) payload_schema_locator
        Remove duplicates preserving ordering
        Filter to remove unneeded entries
        Add all Same_As or Translatable metadata formats
        Add all metadata formats that can be automatically generated

// Return the description of the metadata formats supported for harvest
Build XML results document
EMIT OAI-PMH namespace declarations
        EMIT the required elements
        <responseDate>time of report<responseDate>
        <request
                verb="ListMetadataFormats"          // the literal "ListMetadataFormats"
                identifier=<ID>                     // request ID
                by_doc_ID=<boolean>                 // by document request flag
                by_resource_ID=<boolean>            // by resource request flag
                >
                HTTP_request                        // the HTTP request as a string
        </request>
IF by_doc_ID AND by_resource_ID
        <error code="badArgument" />                // only one can be true
        Complete XML
        EXIT
IF <identifier> provided AND
        by_doc_ID AND
        no resource data description document with doc_ID  = <identifier>
        <error code="idDoesNotExist" />
        Complete XML
        EXIT
IF <identifier> provided AND
        by_resoruce_ID AND
        no resource data description document with resource_locator  = <identifier>
        <error code="idDoesNotExist" />
        Complete XML
        EXIT
IF <identifier> provided AND Results View is empty
        <error code="noMetadaFormats" />
        Complete XML
        EXIT
Build <ListMetadataFormats>
```

```
<ListMetadataFormats>
FOR EACH element in Results View
        <metadataFormat>
                <metadataPrefix>payload_schema</metadataPrefix>
                <schema>payload_schema_locator</schema>
                <metadataNamespace>optionally determine the value for the
                        namespace<metadataNamespace>
        </metadataFormat>
// Add Learning Registry Native JSON format
        <metadataFormat>
                <metadataPrefix>metadataformat structure from the
                Harvest service description</metadataPrefix>
                // the value LR_JSON_0.10.0

        </metadataFormat>
<ListMetadataFormats>
IF <ListMetadataFormats> is empty
        THEN
        DELETE <ListMetadataFormats> element
        <error code=”noMetadaFormats” />
        Complete XML
        EXIT
```

List Sets

The List Sets verb returns the list of sets used to organize resource data descriptions. Support for sets is not defined in this version of the specification. The API SHALL return a standard error indicating that sets are not available.

API

```
        GET<nodeURL>/OAI-PMH?verb=ListSets

        POST <nodeURL>/OAI-PMH
        Post Payload: verb=ListSets

        Request Key-Value Pairs (as per OAI-PMH Specification)
                verb            = ListSets              // literal “ListSets”, required

        Results XML
                Well formed XML instance document that validates according to the OAI-PMH XML XSD
                Contains:
                        <responseDate />                // required XML element
                        <request />                     // required XML element
                        <error />                       // XML element if errors
                        <ListSets/>                     // XML element with results if no errors
```

OAI-PMH: List Sets

```
// Return the description of the sets available for harvest
Build XML results document
EMIT OAI-PMH namespace declarations
EMIT the required elements
        <responseDate>time of report<responseDate>
        <request
                verb=”ListSets”                 // the literal “ListSets”
                >
                HTTP_request                    // the HTTP request as a string
        </request>
// No Set Support
<error code=”noSetHierarchy” />
```

Service Description

```
{
"doc_type":              “service_description”,
▲ "doc_version":                 “0.20.0”,
"doc_scope":             “node”,
"active":                true,
"service_id":            “<uniqueid>”,
"service_type":          “access”,
"service_name":          “OAI-PMH Harvest”,
"service_version":       “0.10.0”,
"service_endpoint":      “<nodeURL>/OAI-PMH”,
▲”service_auth”:                      // service authentication and authorization descriptions
{
▲"service_authz":        [“xxxx”],            // authz values for the service
▲“service_key”:          T/F,          // does service use an access key
▲“service_https”:        T/F           // does service require https
}
"service_data":
 {
  “version”:             “OAI-PMH 2.0”,
  “schemalocation”:      “<XSD URL>”,  // location of the Learning Registry Extended OAI-PMH
                                       // XSD used to validate service responses
▲ “spec_kv_only”:        boolean.      // T to return only spec-defined key-value pairs
                                       // F to return all stored key-value pairs
                                       // optional, default F
  }
}
```

NB: A copy of the schema is currently available at: http://www.learningregistry.org/documents/downloads/OAI-PMH-LR.xsd
There is no guarantee of persistence of this copy of the XSD. A deployed service instance SHOULD use an existing copy of the XSD or maintain a private copy of the XSD according to the node’s data persistence policies.

Basic Query Service

The complete specification of the Basic Query service will be provided in a future draft of the specification.

API

in a future draft of the specification

Basic Query

```
// pseudo code in a future draft of the specification
```

Service Description

```
{
 "doc_type":              “service_description”,
 ▲"doc_version":                   “0.20.0”,
 "doc_scope":             “node”,
 "active":                true,
 "service_id":            “<uniqueid>”,
 "service_type":          “access”,
 "service_name":          “Basic Query”,
 "service_version":       “0.10.0”,
 "service_endpoint":      “<nodeURL>”,
▲”service_auth”:                           // service authentication and authorization descriptions
 {
▲"service_authz":         [“xxxx”],                  // authz values for the service
▲“service_key”:           T/F,             // does service use an access key
▲“service_https”:         T/F              // does service require https
 }
}
```

SRU Service

The complete specification of the SRU/CQL query service will be provided in a future draft of the specification.

API

in a future draft of the specification

SRU

```
// pseudo code in a future draft of the specification
```

Service Description

```
{
 "doc_type":              “service_description”,
 ▲"doc_version":                   “0.20.0”,
 "doc_scope":             “node”,
 "active":                 true,
 "service_id":             “<uniqueid>”,
 "service_type":           “access”,
 "service_name":           “SRU”,
 "service_version":        “0.10.0”,
 "service_endpoint":       “<nodeURL>”,
▲”service_auth”:                            // service authentication and authorization descriptions
 {
▲"service_authz":         [“xxxx”],                  // authz values for the service
▲“service_key”:           T/F,             // does service use an access key
▲“service_https”:         T/F              // does service require https
 }
}
```

SiteMap Service

The complete specification of the XML SiteMap service for web crawlers will be provided in a future draft of the specification.

API

in a future draft of the specification

SiteMap

// pseudo code in a future draft of the specification

Service Description

```
{
 "doc_type":               “service_description”,
 ▲"doc_version":                   “0.20.0”,
 "doc_scope":              “node”,
 "active":                 true,
 "service_id":             “<uniqueid>”,
 "service_type":           “access”,
 "service_name":           “SiteMap”,
 "service_version":        “0.10.0”,
 "service_endpoint":       “<nodeURL>”,
▲”service_auth”:                            // service authentication and authorization descriptions
 {
▲"service_authz":         [“xxxx”],                  // authz values for the service
▲“service_key”:           T/F,             // does service use an access key
▲“service_https”:         T/F              // does service require https
 }
}
```

Atom Pub Service

The complete specification of the Atom Pub service will be provided in a future draft of the specification.

API

in a future draft of the specification

Atom Pub

// pseudo code in a future draft of the specification

Service Description

```
{
"doc_type":           “service_description”,
▲"doc_version":                    “0.20.0”,
"doc_scope":          “node”,
"active":             true,
"service_id":         “<uniqueid>”,
"service_type":       “access”,
"service_name":       “Atom Pub”,
"service_version":    “0.10.0”,
"service_endpoint":   “<nodeURL>”
▲”service_auth”:                  // service authentication and authorization descriptions
 {
▲"service_authz":     [“xxxx”],             // authz values for the service
▲“service_key”:       T/F,         // does service use an access key
▲“service_https”:     T/F          // does service require https
 }
}
```

Administrative Services

Administrative services are used to trigger network node administrative operations, to determine node status or to retrieve descriptive information about a network node. They are used to support monitoring and discovery. Future drafts or versions of this specification MAY define additional administrative services. Future drafts or versions of this specification MAY define additional service query arguments that will customize the returned data.
NB: Provisioning administrative services is optional. They SHOULD NOT be relied on for resource distribution network operations.

Open Question: Do we need to have separate services to return node filters (now part of the general node description) or node connectivity (currently not retrievable).

All administrative services SHALL support HTTP content negotiation. All administrative services SHALL support return of CONTENT-TYPE: text/plain. All administrative services SHOULD support return of text/html, text/xml, application/rdf+xml.

Network Node Status Service

The network node status service is used to return information and operational data about a network node. The service SHALL return all of the key-value pairs listed that have a valid value. The service MAY return additional key-value pairs that indicate status.

A network node SHALL maintain all of the data necessary to return the required key-value pairs.

API

GET <nodeURL>/status

Arguments:

None

Request Object:

None

Results Object:

```
{
 “timestamp”:          “string”,        // time of report, time/date encoding
 "active":             boolean;         // is the network node active
 “node_id”:            “string”,        // ID of the network node
 “node_name”:          “string”,        // name of the network node
 “doc_count”:          integer,         // number of resource data documents
                                        // held by the node
 “install_time”:       “string”,        // time/date of node install
 “start_time”:         “string”,        // server restart time/date
                                        // last reboot
 “last_in_sync”:       “string”,        // time of last inbound sync
                                        // omit if node has not sync’ed
 “in_sync_node”:       “string”,        // id of the node from the last inbound sync
                                        // omit if node has not sync’ed
 “last_out_sync”:      “string”,        // time of last outbound sync
                                        // omit if node has not sync’ed
 “out_sync_node”:      “string”,        // id of the node for the last outbound sync
                                        // omit if node has not sync’ed
 “earliestDatestamp”:  “string”         // oldest timestamp for harvest
                                        // time/date encoding
}
```

Network Node Status

```
// Return the operational status of a network node
DEFINE VIEW on
    network node description document containing the required fields
    + network node operational data containing the required fields
QUERY
TRANSFORM results to specified CONTENT-TYPE
```

Service Description

{

```
 "doc_type":              “service_description”,
▲ "doc_version":                  “0.20.0”,
 "doc_scope":             “node”,
 "active":                true,
 "service_id":            “<uniqueid>”,
 "service_type":          “access”,
 "service_name":          “Network Node Status”,
 "service_version":       “0.10.0”,
 "service_endpoint":      “<nodeURL>”,
▲”service_auth”:                          // service authentication and authorization descriptions
 {
▲"service_authz":        [“xxxx”],               // authz values for the service
▲“service_key”:          T/F,            // does service use an access key
▲“service_https”:        T/F             // does service require https
 }
}
```

Network Node Description Service

The network node description service is used to return descriptive information about a network node, the resource distribution network that it is a part of and the network community that it is a part of. The service SHALL return all of the key-value pairs listed that have a valid value. The service MAY return additional informational values.

API

```
GET <nodeURL>/description

Arguments:
        None

Request Object:
        None

Results Object:
        {
         “timestamp””              “string”,        // time of report, time/date encoding
         "active":                boolean;         // is the network node active
         “node_id”:               “string”,        // ID of the network node
         “node_name”:             “string”,        // name of the network node
         "node_description":      “string”,        // description of the node
        ▲ "node_admin_identity":          “string”,        // identity of node admin
        ▲”node_key”:              “string”,        // node public key
         "network_id":            “string”,        // id of the network
         "network_name":          “string”,        // name of the network
         "network_description":   “string”,        // description of the network
        ▲ "network_admin_identity":       “string”,        // identity of network admin
        ▲”network_key”:                   “string”,        // network public key
         "community_id":          "string" ,         // id of the community
         "community_name":        "string",          // name of the community
         "community_description:"string",                      // description of the community
        ▲ "community_admin_identity":"string",                // identity of community admin
        ▲"community_key":        "string",          // community public key
         "policy_id":             "string",          // id of the policy description
         "policy_version":        "string",          // version identifier for the policy
         "gateway_node":          boolean,           // node is a gateway node
         "open_connect_source":boolean,              // node is willing to be a source
         "open_connect_dest":     boolean,           // node is willing to be a destination
         "social_community":      boolean,           // is community is a social community
         "node_policy":                              // node-specific policies, optional
         {
          "sync_frequency":       integer,           // target time between synchronizations
          "deleted_data_policy":  "string",          // policy value
          ▲ "TTL":                integer,           // minimum time to live for resource data

          ▲"accepted_TOS":        ["string"]                   // list of ToS that the node will accept
        ▲"accepts_anon":         boolean,           // node takes anonymous submissions
        ▲"accepts_unsigned":     boolean,           // node takes unsigned submissions
        ▲"validates_signature":boolean,             // node will validate signatures
        ▲"check_trust":          boolean            // node will evaluate trust of submitter
         }
         "filter":                                   // filter data
         {
          "filter_name":          "string",          // name of the filter
          "custom":               boolean,           // custom filter
          "include_exclude":      boolean,           // accept or reject list
          "filters":                                 // array of filter rules
          [
           {"filter_key":         "string",          // REGEX that matches names
           "filter_value":        "string"           // REGEX that matches values
           }
          ]
         }
        }
```

Network Node Description

```
        // Return the description of a network node
        DEFINE VIEW on
                network node description document containing the required output fields
                + resource distribution network description document containing the required output fields
                + resource distribution network policy document containing the required output fields
                + network community description document containing the required output fields
                + network node filter description document containing the required output fields
        QUERY
```

TRANSFORM results to specified CONTENT-TYPE

Service Description

```
{
 "doc_type":             “service_description”,
▲ "doc_version":                   “0.20.0”,
 "doc_scope":            “node”,
 "active":               true,
 "service_id":           “<uniqueid>”,
 "service_type":         “access”,
 "service_name":         “Network Node Description”,
 "service_version":      “0.10.0”,
 "service_endpoint":     “<nodeURL>”,
▲”service_auth”:                        // service authentication and authorization descriptions
 {
▲"service_authz":        [“xxxx”],              // authz values for the service
▲“service_key”:          T/F,              // does service use an access key
▲“service_https”:        T/F               // does service require https
 }
}
```

Network Node Services Service

The network node services service is used to return the list of services available at a network node. For each service at a node, the service SHALL return all of the key-value pairs listed that have a valid value. The service MAY return additional key-value pairs for a service.

The service SHOULD group and sort the results in some logical form, e.g., by ACTIVE, by TYPE.

API

GET <nodeURL>/services

Arguments:

None

Request Object:

None

Results Object:

```
{
 “timestamp”:            “string”,        // time of report, time/date encoding
 "active":               boolean;         // is the network node active
 “node_id”:              “string”,        // ID of the network node
 “node_name”:            “string”,        // name of the network node
 “services”:                              // array of service description objects
 [
  {"active":             boolean;         // is the service active
```

```
                "service_id":          “string”,        // id of the service
                "service_type":        “string”,        // fixed vocabulary
                "service_name":        “string”,        // name of the service
                "service_description": “string”,        // description of the service
                "service_version":     “string”,        // version number of the service description
                "service_endpoint":    “string”,        // URL of service
               ▲”service_auth”:                         // service authentication and authorization
descriptions
               {
               ▲"service_authz":       [“string”],      // authz values for the service
               ▲“service_key”:                boolean,         // does service use an access key
                       ▲“service_https”:      boolean          // does service require https
               }
                "service_data":        {}               // service-specific name-value pairs
                }
               ]
              }
```

Network Node Services

```
    // Return the description of network node services
    DEFINE VIEW on
        network node description document containing the required output fields
        + ALL network node service description documents containing the required output fields
        GROUPED and ORDERED on service attributes.
    QUERY
    TRANSFORM results to specified CONTENT-TYPE
```

Service Description

```
    {
    "doc_type":             “service_description”,
  ▲ "doc_version":                  “0.20.0”,
    "doc_scope":            “node”,
    "active":               true,
    "service_id":           “<uniqueid>”,
    "service_type":         “access”,
    "service_name":         “Network Node Services”,
    "service_version":      “0.10.0”,
    "service_endpoint":     “<nodeURL>”,
  ▲”service_auth”:                  // service authentication and authorization descriptions
    {
  ▲"service_authz":         [“xxxx”],       // authz values for the service
  ▲“service_key”:           T/F,        // does service use an access key
  ▲“service_https”:         T/F         // does service require https
    }
  }
```

Resource Distribution Network Policy Service

The resource distribution network policies service is used to return information about the policies that apply to the resource distribution network that the network node is a part of. The service SHALL return all of the key-value pairs listed that have a valid value. The service MAY return additional policy key-value pairs. The service MAY be called at any node in the resource distribution network; all network nodes store an identical copy of the policy data.

API

```
GET <nodeURL>/policy

Arguments:
        None

Request Object:
        None

Results Object:
        {
         “timestamp”:            “string”,       // time of report, time/date encoding
         "active":               boolean;        // is the network node active
         “node_id”:              “string”,       // ID of the network node
         “node_name”:            “string”,       // name of the network node
         "network_id":           “string”,       // id of the network
         "network_name":         “string”,       // name of the network
         "network_description":  “string”,       // description of the network
         "policy_id":            “string”,       // id of the policy description
         "policy_version":       “string”,       // version identifier for the policy
         “TTL”:                  integer         // minimum time to live for resource data
        }
```

Resource Distribution Network Policy

```
// Return the description of network policies
DEFINE VIEW on
        network node description document containing the required output fields
        + resource distribution network description document containing the required output fields
        + resource distribution network policy document containing the required output fields
QUERY
TRANSFORM results to specified CONTENT-TYPE
```

Service Description

```
{
 "doc_type":            “service_description”,
▲ "doc_version":                “0.20.0”,
 "doc_scope":           “node”,
 "active":              true,
 "service_id":          “<uniqueid>”,
 "service_type":        “access”,
 "service_name":        “Resource Distribution Network Policy”,
 "service_version":     “0.10.0”,
 "service_endpoint":    “<nodeURL>”,
▲”service_auth”:                            // service authentication and authorization descriptions
 {
▲"service_authz":       [“xxxx”],                    // authz values for the service
▲“service_key”:                 T/F,                 // does service use an access key

▲“service_https”:       T/F                 // does service require https
 }
}
```

Broker Services

Broker services operate at a network node to process, transform, augment, amplify or otherwise manipulate the resource data held at a node. No broker services are currently defined. Broker services will be defined in future drafts or versions of the specification.

Common Data Model and API Attributes and Behaviors

The definition of several common attributes shared across all data models and APIs, along with common API behaviors are specified here as a single point of specification. In case of a discrepancy, the definition here SHALL take precedence over the definition elsewhere in this specification.

Data Model Attributes

Identifiers

Most data models include one or more *identifiers*. An identifier SHALL be unique within a defined scope or context. Unless otherwise specified, the scope for all identifiers SHALL be all implementations of the Learning Registry. Unless otherwise specified by an implementation, an identifier SHALL conform to ISO/IEC 11578:1998, ISO/IEC 9834-8, RFC 4122, and SHOULD use Version 5 (SHA-1 Hash). These specifications standardize the generic OSF DCE UUID. As a data type, an identifier is commonly encoded as a string.

NB: What is called an identifier in a data model is more precisely just a label. The use of the label to identify an instance of the data model within the scope or context of the Learning Registry makes it an identifier (within that scope).

NB: For many items, the scope could be all implementations of the Learning Registry within one network community. Extending the scope to all implementations is an intentional simplification.

Open Question: UUID Version 1 (MAC Address) Version 5 (SHA-1 Hash)?

Strings and Internationalization

All character strings SHALL support full UTF-8 encoding of Unicode character representation.
▲*NB*: JSON strings default to UTF-8 encoding. ECMAScript strings are UTC-16.

Time and Date

The format for all times and dates SHALL conform to ISO 8601. All encoded dates and times SHALL include a Time Zone designator. All stored times SHALL be expressed to at least second precision with a known timezone. More precise values MAY be used.
NB: Some data models and APIs require times be UTC 0 and may place additional restrictions on times and dates.
ToDo: Specify JSON encoding.

API Attributes and Behaviors

Description here about RESTful APIs, CoolURIs, Context negotiation, application/JSON vs text/plain

HTTP requests SHALL use HTTP 1.1. Communications MAY use TLS.

HTTP requests SHOULD include a CONTENT-TYPE header. Unless noted, the header SHOULD be CONTENT-TYPE: application/JSON

HTTP responses SHOULD include a CONTENT-TYPE header. Unless noted, the header SHOULD be CONTENT-TYPE: text/plain; charset=utf=8

Transactional Behaviors and Data Consistency

Unless stated in an individual API specification, transactional atomicity SHALL BE document granularity.

Requirements for consistency of documents across multiple nodes apply only when the nodes are consistent. Prior to, or during document distribution, documents MAY be inconsistent.
NB: The distribution model assumes the underlying system SHALL produce *eventual consistency*.

Resource Data Validation and Publication

All resource data publishing and distribution services SHALL validate all documents before the document is stored at the node.

- All required fields SHALL be present.
- Only mutable fields MAY be changed in an update.
- The node SHALL reject a submission where the payload does not correspond with the declared payload.
- The node MAY validate an attached or linked payload.
- The node MAY validate an inline payload.
- Prior attachments SHALL be deleted in an update.
- Default values SHALL be filled in.

- Node-specific fields SHALL be filled in.

Resource Data Validation and Publication

```
// Validate a resource data description document
// is the submission well formed
IF any required element is missing
    THEN
        REJECT the document
        EXIT
// changes in mutable fields are only allowed in an update
IF submission is an update
    IF any immutable field in the new document does not match old field
        THEN
            REJECT the document
            EXIT
// does the payload match the declaration
IF payload_placement = “linked” and no payload_locator provided
    THEN
        REJECT the document
        EXIT
IF payload_placement = “inline” and no resource_document in the submission
    THEN
        REJECT the document
        EXIT
IF payload_placement = “attached” and no attachment
    THEN
        REJECT the document
        EXIT
// payload must match the schema and validate
IF payload_schema does not correspond to resource_data_type
    THEN
        REJECT the document
        EXIT
VALIDATE the payload
// updates invalidate existing attachments
IF submission is an update
    THEN delete any attachments
// Generate the ID if required
IF doc_ID isn’t provided
    THEN generate a doc_ID
// Set local node data
publish_node := node_id
IF submission is an update
    THEN
        update_timestamp := node_timestamp := current time  // granularity of seconds
    ELSE
        create_timestamp :=update_timestamp := node_timestamp := current time
```

```
IF frbr_level not specified
    THEN frbr_level := “copy”
```

Open Question: Should an update delete the attachments automatically, or should this be an option?

Resource Data ToS, Signatures and Trust Policy Enforcement

▲All resource data publishing services and resource data distribution services MAY apply ToS, digital signature and submitter identity checks to resource data.

- The node MAY reject an anonymous submission or any other submission according to its policy.
- The node MAY reject a submission from an untrusted submitter.
- The node MAY reject a submission without a known terms of service.
- The node MAY reject a submission that is not signed.
- The node MAY reject the submission if the signature cannot be verified.

NB: The acceptable node policies and terms of service are not defined in this specification. The specification requires that ToS, digital signatures and trust checks be performed according to node policies. The outcome of those checks, and the actions taken, are governed by node, network or community-specific policies that are out of scope for this specification.

▲*NB*: Declared policies SHOULD be applied consistently in both publication and distribution.

▲*NB*: A node MAY apply other policies or MAY apply policies without declaring them in the node description.

▲*NB*: An implementation MAY check policies in any order. It MAY evaluate all policies or do a short-circuit evaluation and stop when any policy violation is found.

Resource Data ToS, Signatures and Trust Policy Enforcement

```
// Check Policies
▲IF the service applies ToS checks
    AND the resource data description document TOS is unacceptable
    THEN // indicate ToS was rejected
        REJECT the document
        EXIT
▲IF the service does not accept anonymous submissions
    AND the resource data description document has submitter_type==”anonymous”
    THEN // indicate submitter type was rejected
        REJECT the document
        EXIT
▲IF the service validates the submitter or submitter trust
    AND the resource data description document submitter cannot be verified or trusted
    THEN // indicate submitter was rejected
        REJECT the document
        EXIT
▲IF the service requires a signature
    AND the resource data description document signature not present
    THEN // indicate signature was rejected
        REJECT the document
        EXIT
```

```
▲IF the service validates the signature
    AND the resource data description document signature cannot be verified
    THEN // indicate signature was rejected
        REJECT the document
        EXIT
```

Resource Data Filtering

All resource data publishing services and resource data distribution services apply filters to resource data. If a network node filter document is stored at a node, the filter SHALL be applied to a resource data description document before the document is stored at the node.

Either a custom filter or expression-based filters MAY be defined. If there is a custom filter (expressed in custom code at the node), expression-based filters SHALL be ignored. A custom filter SHOULD NOT be used when the filters can be expressed using expression-based filters.

A filter defines either the resource data documents that pass the filter (and are stored; all other resource data documents are not stored), or resource data documents that are rejected by the filter (and are not stored, all other documents are stored).

An expression-based filter contains a list of regular expressions that are used to match keywords/names in the resource data description document, and a regular expression that is used to match values for the keywords. If the filter key matches any keyword/name in the resource data, and if any value for that key in the resource data matches the filter value, the filter is successful, i.e., for an “include” filter, the document is included; for an exclude filter, the document is “excluded”. Matching is an “or”.   A successful match short circuits further matching.

The filter SHALL be applied against all top-level elements in the resource data description document. Behavior for filtering against linked resource data, attachments or the inline resource data is not currently defined.

Resource Data Filtering

```
// Filter a resource data description document
// No filter test
IF the network node filter description document does NOT exist
    THEN store the resource data description document
        EXIT
// Filter not active test
IF NOT active in the network node filter description document
    THEN store the resource data description document
        EXIT
// Use custom filter if defined
IF custom_filter in the the network node filter description document
    THEN eval the custom filter code
        IF the code returns true
            THEN store the resource data description document
```

```
                    EXIT
// Expression-based filtering
// Does the filter match the document
match := F
FOR EACH filter in the network node filter description document
          FOR EACH key/name in the resource data description document
          IF the filter_key REGEX matches the key/name
                    IF the filter_value is NULL
                              THEN match := T
                                        SKIP ALL
                    FOR EACH value of the  key/name in the resource data description document
                              IF the filter_value REGEX matches the value
                                        THEN match := T
                                                  SKIP ALL
// Store or reject
IF include_exclude
          IF match  // matches what to include
                    THEN  store the resource data description document
                              EXIT
                    ELSE EXIT // don’t store
          IF NOT match // doesn’t match what to exclude
                    THEN  store the resource data description document
                              EXIT
                    ELSE EXIT // don’t store
```

Operations

This section outlines one approach of how to use the specified network model, data models and APIs to set up and operate a Learning Registry network community.

Networks and Communities

Building the Network Description

Node-specific information (i.e., description, filters, services and connectivity) SHALL be maintained on a per node basis. Authorized document updates MAY be applied only at the node. All node-specific information MAY be maintained in a node document collection per node.

Network-specific information (i.e., network description, network policy) SHALL be replicated on a per node basis. The initial document MAY be stored at any node in the network. Authorized document updates MAY be applied at any node in the network. Replication, using the node’s connectivity information, MAY be used to propagate the documents throughout the network. All network-specific information MAY be maintained in a network document collection per node.

Community-specific information (i.e., community description) SHALL be replicated on a per node basis. The initial document MAY be stored at any node in the network. Authorized document updates MAY be applied at any node in the community. Replication, using the node’s connectivity information, MAY be used to propagate the documents throughout the network, including across network gateways between different communities. All community-specific information MAY be maintained in a community document collection per node.

The overall network description MAY be inconsistent when publishing individual documents that describe the network. When using document propagation, the node's document collections MAY be out of sync, temporarily violating the requirement for identical values at all nodes. Eventual consistency SHALL be enforced. *NB*: Need to determine if lack of sync can introduce any security holes.

An implementation MAY place the node-specific information in one document database that is not replicated, network-specific information in a second database that is replicated throughout the network, and community-specific information in a third database that is replicated across the community. Resource data is stored in another database that is replicated throughout the community using the defined connectivity.

Except for a optional local node storage used to maintain private state that is not replicated, other document databases SHALL NOT be defined for the purpose of holding node-specific, network-specific or community-specific information.

Open Issue: As defined, gateways permit networks that allow documents to be replicated **into** the network from another network, but do not permit documents to be replicated **out** of the network (unless there is a corresponding bi-directional link). With such a topology, community-specific information cannot reach all nodes if it is published to a node within a network with no outbound flow; the information will not reach other networks. Possible solutions:

- publish the community-specific information to a node in each network that does not have outbound connectivity.
- define a distribution strategy (and possible extensions of node connectivity information) to allow bi-directional flow across network gateways or that limit flow based on document type.
- define node connections that provide only for distribution of network or community information, not resource data.

Likewise, gateways transmit documents across network boundaries. They do not limit distribution to be within a single network. Thus network-specific documents can cross network boundaries. Possible solutions:

- define a distribution strategy that is limited to a network and not across gateways.
- define node connections that provide only for distribution of network information.

NB: The procedures described below are currently incomplete and do not implement any of these solutions. They call for "network wide" (meaning network restricted) or "community wide" distribution.

Open Question: How to establish a security model so that only an authorized user may update a description at any node. Is it necessary to sign the descriptions or to store a public key in a description?

Open Question: How to constrain the network so that if someone finds a connection, they cannot exploit it.

The network model MAY be instantiated in a set of procedures. The procedures assume the *resource distribution network description* document, *resource distribution network policy* document and *network community description* document are published before the network is expanded from the base node. If network expansion is done prior to these documents being created, appropriate *Distribution* processes SHALL be triggered after the documents are published to distribute them to all nodes in the network or community.

NB: The procedures only establish the network structure and descriptive documents. They do not populate the network with resource data. In particular, adding a new node to a network does not load resource data into the node's resource data description document database.

NB: The procedure does not state what services need to be provisioned at a node. Any organization can provision their own nodes, determine what services to offer (under what access constraints) and add the node to the network and community. To participate in resource distribution, each source and target resource data distribution node SHALL provision the appropriate source or destination process of the resource distribution service.

The procedures include VALIDATE steps which are explicit semantic rules that need to be enforced, generally involving values in different documents. The procedures also include REJECT steps. These are general rules designed to check if a document is well formed, that all mandatory fields are present, that values are from constrained vocabularies, that immutable values are not changed on update, etc. The procedure exits without changing the state of any database if validation fails (i.e., the procedure is a transaction).

Establish a Network Node

```
// Create one new node
FIND  a source for the node software, e.g., the Learning Registry GitHub
    // out of band process
INSTALL the node software on a hardware node // physical or virtual
CONFIGURE  the node software
CREATE an identity for the node owner
    // used to own node documents, proxy for document owner
CREATE a network node description document for the node
PUBLISH the network node description document to the new node
    by the node owner
    to the node's node document database
    REJECT if the network node description document is not valid
    REJECT if a network node description document exists
IF the node filters published or distributed data
    CREATE a network node filter description document for the node
    PUBLISH the network node filter description document to the new node
        by the node owner
        to the node's node document database
        REJECT if the network node filter description document is not valid
```

```
            REJECT if a network node filter description document exists
    FOR EACH service that the node provides:
        CREATE a network node service description document for the service
        PUBLISH the network node services description document  to the node
            by the node owner
            to the node’s node document database
        REJECT if the network node service description document is not valid
```

Establish a Resource Distribution Network

```
    // Create a base, one-node network with a network description
    PREREQUISITE: one active node // denoted the base node
        // via the Establish a Node process
    CREATE an identity for the network owner
    CREATE a resource distribution network description document
    PUBLISH the resource distribution network description document to the first node
        by the owner of the resource distribution network description document
        to the node’s network document database
        VALIDATE // same network
            network_id in the network node description document =
            network_id in the resource distribution network description document
        REJECT if the resource distribution network description document is not valid
        REJECT if a resource distribution network description document exists
    CREATE a resource distribution network policy document
    PUBLISH the resource distribution network policy document to the base node
        by the owner of the resource distribution network policy document
        to the node’s network document database
        VALIDATE // same network
            network_id in the network node description document =
            network_id in the resource distribution network policy document
        REJECT if the resource distribution network policy document is not valid
        REJECT if a resource distribution network policy document exists
```

Establish a Network Community

```
    // Create a base, one-network community with a network description
    PREREQUISITE: one active node // denoted the base node
        // via the Establish a Network Node process
    PREREQUISITE: one active network
        // via the Establish a Resource Distribution Network process
    CREATE an identity for the network community owner
    CREATE a network community description document
    PUBLISH the network community description document to the base node
        by the owner of the network community
        to the node’s network community document database
        VALIDATE // same community in network and community descriptions
            community_id in the resource distribution network description document =
            community_id in the network community description document
        VALIDATE // same community in node and community descriptions
            community_id in the network node description document =
            community_id in the network community description document
        REJECT if the network community description document is not valid
        REJECT if a  network community description document exists
```

Add a Network Node to a Resource Distribution Network

```
// Add a node to an existing network
// NB: Nothing in this process loads resource data into the node
PREREQUISITE: an established active network // one or more active nodes
PERFORM the Establish a Network Node process to create a node // denoted the new node
FIND another active node in the network to connect to // denoted the existing node
    // discovery and agreement to connect is out of band
PERFORM the Adding Connections within a Resource Distribution Network process
    source node is the existing node
    destination node is the new node
PERFORM the Distribution process to replicate the network document database
    from the source node to the destination node
    // propagates network description and policy to only the new node
    // may proceed asynchronously
```

Adding Connections within a Resource Distribution Network

```
// Add connectivity between two existing nodes in a network
// A source node connects to a destination node
PREREQUISITE:  an established network with two or more nodes
PREREQUISITE:  the active source node is known (connecting node)
PREREQUISITE:  the active destination node is known (connected node)
    // discovery and agreement to connect is out of band
VALIDATE // same network
    network_id in the source node's network node description document =
    network_id in the destination node's network node description document
VALIDATE // same community
    community_id in the source node's network node description document =
    community_id in the destination node's network node description document
CREATE a network node connectivity description document with
    source_node_url := source node URL
    destination_node_url := destination node URL
    gateway_connection := F
PUBLISH the network node connectivity description document
    by the source node owner
    to the source node's node document database
REJECT if the network node connectivity description document is not VALID
REJECT if the same source -> destination active connection exists
```

Connect Networks within a Community

```
// Add a gateway connection between two networks
// NB: Nothing in this process distributes resource data across the gateway
PREREQUISITE:  two established networks
PREREQUISITE:  the  active source node is known (connecting node)
```

```
PREREQUISITE:  the active destination node is known (connected node)
    // discovery and agreement to connect is out of band
VALIDATE // different network
    network_id in the source node’s network node description document <>
    network_id in the destination node’s network node description document
VALIDATE  // same community
    community_id in the source node’s network node description document =
    community_id in the destination node’s network node description document
VALIDATE // no gateway
    FOR EACH source node’s network node connectivity description document
        NOT gateway_connection
CREATE a network node connectivity description document with
    source_node_url := source node URL
    destination_node_url := destination node URL
    gateway_connection := T
PUBLISH the network node connectivity description document
    by the source node owner
    to the source node’s node document database
REJECT if the network node connectivity description document is not VALID
REJECT if the same source -> destination active connection exists
```

Connect Communities

```
// Add a gateway connection between two communities
// NB: Nothing in this process distributes resource data across the gateway
PREREQUISITE:  two communities networks
PREREQUISITE:  the active source node is known (connecting node)
PREREQUISITE:  the active destination node is known (connected node)
    // discovery and agreement to connect is out of band
VALIDATE // social communities
    social_community T in the source node’s network community description document
    social_community T in the destination node’s network community description document
VALIDATE // different network
    network_id in the source node’s network node description document <>
    network_id in the destination node’s network node description document
VALIDATE  // different community
    community_id in the source node’s network node description document <>
    community_id in the destination node’s network node description document
VALIDATE // no gateway
    FOR EACH source node’s network node connectivity description document
        NOT gateway_connection
CREATE a network node connectivity description document with
    source_node_url := source node URL
    destination_node_url := destination node URL
    gateway_connection := T
PUBLISH the network node connectivity description document
    by the source node owner
    to the source node’s node document database
REJECT if the network node connectivity description document is not VALID
```

```
    REJECT if the same source -> destination active connection exists
```

Maintaining Networks and Communities

An established network or community MAY be maintained by updating descriptions of network nodes, their services, their connectivity, descriptions of resource distribution networks and of network communities. By definition, elements of the network model SHALL NOT be deleted; they are transitioned from active to not active. *NB:* The data models contain forward links to other models. Deleting a document would require network-level garbage collection to determine when all links to a document have been deleted.

If updating the description of a network node, its services or connectivity, the description of a the distribution network or the network community causes the descriptions to violate the requirements for the network description, the node SHALL NOT be considered to be a part of the corresponding distribution network and network community and SHALL NOT participate in any network or resource operations.

Change Network Node Description

```
    // update the description of any node
    PUBLISH the updated network node description document to the node
        by the node owner
        to the node's node document database
        REJECT if the network node description document is not valid
        REJECT if the network node description document is not an update
        // node may have only one network node description
```

Delete a Network Node

```
    // remove a node from a network
    // but the node remains, inaccessible
    // first sync the documents so that others have them
    // sync before delete is an operational policy choice; could be modeled in policy
    COMMIT all outstanding resource data description database operations
    PERFORM the Distribute Resource process
    FOR EACH resource data description document
        delete the document from the node's resource data description document database
        // this is an explicit delete
    PUBLISH the updated network node description document to the node
        by the node owner
        ACTIVE = F
        to the node's node document database
        REJECT if the network node description document is not valid
        REJECT if the network node description document is not an update
        // node may have only one network node description
    FOR EACH network node services description document
        PERFORM the Delete Node Service process
    FOR EACH network node connectivity description document
        PERFORM the Delete Node Network Connection process
```

Change Node Service Description

```
    // update the description of a service at any node
    PUBLISH the updated network node services description document  to the node
        by the node owner
        to the node’s node document database
    REJECT if the network node service description document is not valid
    REJECT if  the network node service description document is not an update
```

Add Node Service

```
    // add a service to any node
    PUBLISH the new network node services description document  to the node
        by the node owner
        to the node’s node document database
    REJECT if the network node service description document is not valid
    REJECT if the network node service description document is not an addition
```

Delete Node Service

```
    // delete a service from any node
    PUBLISH the updated network node services description document to the node
        ACTIVE = F
        by the node owner
        to the node’s node document database
    REJECT if the network node service description document is not valid
    REJECT if the network node service description document is not an update
```

Change Node Network Connectivity

```
    // change the connectivity description of a connection from a node
    // unless there are mutable extension data elements, the process is a NO-OP
    // all other data elements are immutable
    PUBLISH the updated network node connectivity description document
        by the source node owner
        to the source node’s node document database
    REJECT if the network node connectivity description document is not valid
    REJECT if the network node connectivity description document is not an update
```

Delete Node Network Connectivity

```
    // remove the connection from a node to another node
    // applies to intra-network or inter-network or inter-community
    PUBLISH the updated network node connectivity description document
        ACTIVE = F
        by the source node owner
        to the source node’s node document database
    REJECT if the network node connectivity description document is not valid
    REJECT if the network node connectivity description document is not an update
```

Change Node Filters

```
    // change the filters at a node
    PUBLISH the updated network node filter description document
```

```
        by the source node owner
        to the source node’s node document database
    REJECT if the network node filter description document is not valid
    REJECT if the network node filter description document is not an update
```

Delete Node Filters

```
    // delete ALL filters at a node
    PUBLISH the updated network node filter description document
        ACTIVE = F
        by the source node owner
        to the source node’s node document database
    REJECT if the network node filter description document is not valid
    REJECT if the network node filter description document is not an update
```

Change Resource Distribution Network Description

```
    // change the resource distribution network description
    // applied at some node
    PUBLISH the updated resource distribution network description document
        by the resource distribution network description document owner
        to the node’s network document database
    REJECT if the resource distribution network description document is not valid
    REJECT if the resource distribution network description document is not an update
    PERFORM a network-wide Distribution process to replicate the network document database
        to the other nodes in the network
        // propagates resource distribution network description to all nodes in the network
        // may proceed asynchronously
```

Delete Resource Distribution Network Description

```
    // delete the resource distribution network description
    // applied at some node
    PUBLISH the updated resource distribution network description document
        ACTIVE = F
        by the resource distribution network description document owner
        to the node’s network document database
    REJECT if the resource distribution network description document is not valid
    REJECT if the resource distribution network description document is not an update
    PERFORM a network-wide Distribution process to replicate the network document database
        to the other nodes in the network
        // propagates resource distribution network description to all nodes in the network
        // may proceed asynchronously
```

Change Resource Distribution Network Policy

```
    // change the resource distribution network policy
    // applied at some node
    PUBLISH the updated resource distribution network policy document
        by the resource distribution network policy document owner
        to the node’s network document database
    REJECT if the resource distribution network policy document is not valid
    REJECT if the resource distribution network policy document is not an update
    PERFORM a network-wide Distribution process to replicate the network document database
        to the other nodes in the network
        // propagates resource distribution network policy to all nodes in the network
        // may proceed asynchronously
```

Delete Resource Distribution Network Policy

```
    // delete the resource distribution network policy
    // applied at some node
    PUBLISH the updated resource distribution network policy document
        ACTIVE = F
        by the resource distribution network policy document owner
        to the node's network document database
    REJECT if the resource distribution network policy document is not valid
    REJECT if the resource distribution network policy document is not an update
    PERFORM a network-wide Distribution process to replicate the network document database
        to the other nodes in the network
        // propagates resource distribution network policy to all nodes in the network
        // may proceed asynchronously
```

Change Network Community Description

```
    // change the network community description
    // applied at some node in some network in the community
    // node must have connectivity to reach all other networks
    // otherwise apply to multiple nodes
    PUBLISH the updated network community description document
        by the network community description document owner
        to the node's network community document database
    REJECT if the network community description document is not valid
    REJECT if the network community description document is not an update
    PERFORM a community-wide Distribution process to replicate the
        community document database to the other nodes in the community
        // propagates community description to all nodes in the community
        // may proceed asynchronously
```

Delete Network Community Description

```
    // delete the network community description
    // applied at some node in some network in the community
    // node must have connectivity to reach all other networks
    // otherwise apply to multiple nodes
    PUBLISH the updated network community description document
        ACTIVE = F
        by the network community description document owner
        to the node's network community document database
    REJECT if the network community description document is not valid
    REJECT if the network community description document is not an update
    PERFORM a community-wide Distribution process to replicate the
        community document database to the other nodes in the community
        // propagates community description to all nodes in the communities
        // may proceed asynchronously
```

Network Discovery

Finding all the nodes in a network or community is a non core service. One approach is to use replication and distribution to build a complete list of all network links at each node (an alternative is to traverse the network and build the link structure). Given a database of node-specific documents that includes a node's connectivity that is not replicated (private to the node), and a second duplicate database that is replicated across the entire network or community, first replicate the connectivity document from the private node database to the network or community database stored at the node. This replication is done at each node in the network. It is a one-way replication from the private node database to the second database, not a full synchronization. Then distribute (synchronize) the second database across all nodes. Distribution (synchronization) across nodes of the network or community database will build a network connectivity link table at each node.

The completeness of the table in showing the entire network or part of the network will depend on the connectivity and gateways, i.e., the connectivity of a network will not be propagated to nodes outside the network unless there is a directional gateway. The tables of networks or communities with only inbound connectivity will include the entries of the external networks.

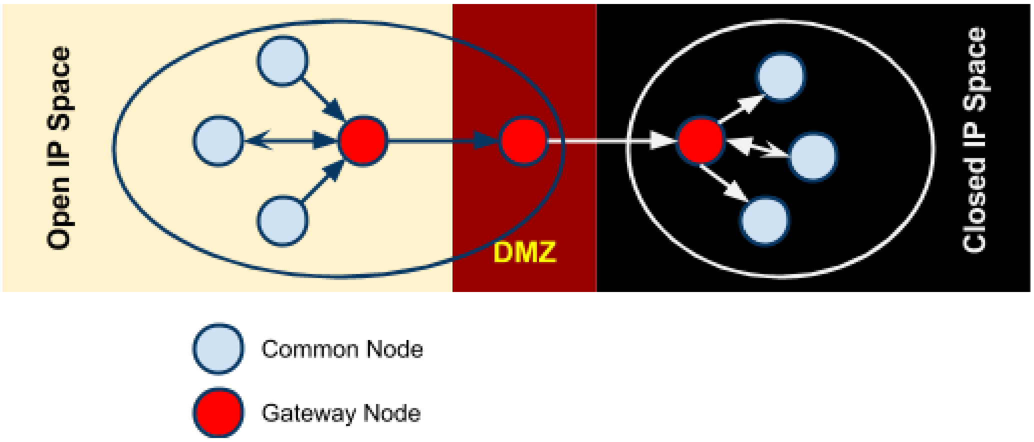
NB: This process only builds the network link table at a point in time. It does not provide real-time connectivity status.

Network Design

Any resource distribution network is technically open; any node MAY connect. Managing connectivity and details of the topology and operations of any specific resource distribution network (and its connection to other resource distribution networks and network communities) SHALL be determined by the governance and policy rules of a resource distribution network or network node, not via this specification.

Multiple resource distribution networks and network communities MAY be established and connected. In addition to external governance and policies, this specification imposes restrictions on connections between network communities and resource distribution networks, i.e., 1:1 connections only using gateway nodes. Because of the technical burden imposed, multiple resource distribution networks and network communities SHOULD only be created when these specification-imposed criteria are needed to enable additional security and technical criteria that cannot be adequately satisfied by policy alone.

For example, connecting an open community in an open IP space to a more restricted community might involve establishing a gateway node in the open community in a public IP space, providing a gateway node in for the restricted community in a DMZ, opening limited network ports between the DMZ and the other nodes in restricted community that operate within a closed IP space (possibly using an additional gateway), and locking down the connections between the open community and the DMZ gateway node. The illustration shows an example of such a gateway structure.



Resource Data Publication, Distribution and Access

Once a resource distribution network has been established, external agents use the publish services at network nodes to publish or update resource data to nodes in the resource distribution network. Network nodes SHALL store the valid, published resource data in their resource data description documents database. Publication MAY be performed via any of the supplied publish services, value-added services or via tools and applications that are built upon these services. Other mechanisms to publish resource data (e.g., direct access to the underlying data stores) SHALL NOT be supported.

Publication SHALL satisfy trust and identity conditions. The same resource data may be published to one or more nodes in one or more resource distribution networks subject to their policies and the trust and identity conditions.

Publication SHALL satisfy network node filters. Only resource data that passes the filters (if they exist) SHALL be published. Filters are optional.

Publication SHALL add or update any node-specific data.

In distribution, destination nodes SHALL filter all incoming resource data. Only resource data that passes the filters (if they exist) SHALL be persisted. Filters are optional.

Distribution SHALL add or update any node-specific data.

Resource distribution network nodes distribute their data to other nodes. If a network node provides the resource data distribution service, it SHALL periodically launch the resource data distribution service to distribute resource data from the node to its connected nodes. Any node that wants to establish a connection with another node, i.e., wants to be the target of distribution, SHALL support the necessary services that allow the source node to distribute data to it.

Resource data is available for access by external agents through the access services at network nodes.

Access MAY be provided via any of the supplied access services, value-added services or via tools and applications that are built using these services. Other mechanisms to access resource data (e.g., direct access to the underlying data stores) SHALL NOT be supported.

Access SHALL satisfy trust and identity conditions. When available via distribution, the same resource data may be accessed from one or more nodes in one or more resource distribution networks, subject to their policies and the trust and identity conditions.

Resource distribution network nodes MAY provide broker services. Operation of a broker service SHALL be determined by the governance and policy rules of a resource distribution network or network node, not via this specification.

Service provisioning (publish, access, distribution, broker) SHALL be determined by the governance and policy rules of a resource distribution network or network node, not via this specification.

Resource Data Persistence

A node SHALL persist resource data stored at a node for at least as many days as specified by the TTL (time to live) in the resource distribution network policy description. A node MAY "delete" documents that have been stored longer than the network's TTL.

If the node supports harvest, the node SHALL maintain a value for earliestDatestamp. This value is the oldest publish, update or delete time for a resource data description document that can be accessed at the node via harvest. The timestamp SHALL be precise to the nearest second. The value is based on the time the resource data description documents are published at the node (node_timestamp).

Access to information about the deletion of resource data is governed by a node-specific policy, deleted_data_policy:

- no -- the node SHALL not *expose* any data about deleted resource data description documents. The node MAY maintain information about deletions.
- persistent -- the node SHALL maintain and expose data about deleted resource data description documents. Information about deleted resource data SHALL be persisted as long as the node remains active, i.e., information about deletion is not governed by the TTL.
- transient -- the node MAY maintain and expose data about deleted resource data description documents. The node MAY establish any policy for how long or if it maintains and exposes data about deleted resource data description documents.

NB: This is the information about the deletion, not the actual deletion. What it means to "delete" a document is not specified, e.g., it is removed or just marked as deleted. Tracking of deletions MAY be independent of the actual TTL and the actual deletion of the resource data.
Open Question: Should the data persistence policies be network, not node, specific.

Open Issue: Using pure replication, when one node deletes a document, the delete will propagate. This MAY invalidate the required information about the tracking and persistence of deletions. What is the model for deletion versus TTL and access to deletion status for harvest?

Network Administration

Resource distribution network nodes MAY provide administrative services. Operation of an administrative service SHALL be determined by the governance and policy rules of a resource distribution network or network node, not via this specification.

Provisioning of administrative services SHALL be determined by the governance and policy rules of a resource distribution network or network node, not via this specification.

Open Question: Do we need to make any administrative services mandatory? ▲No -- all services are optional
Open Question: Do we need a mechanism to control access to network data models beyond authentication controls?

Glossary

The following terms are used in this document as defined.
Additional terms will be provided in a future draft or version of the specification.

access (v): to obtain resource data from a network node by an agent that is external to a resource distribution network.

broker (n): a server process that provides transformative or data amplification processing of resource data.

community (n): see *network community*.

common node (n): a network node in a resource distribution network that may provide any service to process resource data and that may connect to any other node in the same resource distribution network for the distribution of resource data within the resource distribution network.

distribute (v): to copy or synchronize resource data from one network node to another.

gateway node (n): a network node in a resource distribution network that provides an interconnection to a network node in a different resource distribution network (either in the same network community or in a different network community) for the distribution of resource data across the network boundary.

harvest (v): to access a network node and obtain sets of resource data; the accessing agent is the harvestor; the network node is the harvestee. Harvest is typically based on timestamps used to identify new resource data held at the harvestee.

identifier (n): the name (i.e., a label [e.g., a string] in an authoritative context) associated with a thing (anything that can be given an identifier).

learning resource (n): any (digital) resource that is designed for, or has been used, in an educational context.

metadata (n): formally authored and curated information describing a learning resource. Also denoted *first party* metadata.

network (n): see *resource distribution network.* A network need not correspond to a physical or logical network of computing devices.

network community (n): a group of interconnected resource distribution networks.

network node (n): a service end point in a resource distribution network that may provide services to process resource data and that may connect to any other nodes to distribute resource data. A network node need not correspond to a physical or logical computing device.

node (n): see *network node*.

paradata (n): information describing the contextual use of a learning resource. It includes informally authored information and data obtained directly through monitoring the use of a learning resource, its metadata or its paradata. Also denoted *second party* metadata.

publish (v): to submit resource data to a network node from a source external to the node's resource distribution network.

pull (v): to distribute resource data from A to B, initiated by B.

push (v): to distribute resource data from A to B, initiated by A.

resource (n): see *learning resource.*

resource data (n): any data that describes a learning resource, including, but not limited to metadata and paradata.

resource distribution network (n): a group of interconnected network nodes that operate under an agreed set of policies.

service (n): a process applied to resource data or system descriptive and operational data operating on a network node.

References

References below contain both normative and informative references. Unless otherwise noted, this specification references specific versions of other normative standards. More recent versions SHALL NOT be used.

Additional references will be provided in a future draft or version of the specification.

- CoolURIs 2008: *Cool URIs for the Semantic Web*, http://www.w3.org/TR/cooluris/
- DC 1.1: Dublin Core Metadata Element Set, Version 1.1, http://dublincore.org/documents/dces/
- FRBR: *Functional Requirements for Bibliographic Records,* International Federation of Library Associations and Institutions, 1998, ISBN: 359811382X, http://www.ifla.org/VII/s13/frbr/frbr.pdf
- ▲*GNU Privacy Handbook*, http://www.gnupg.org/gph/en/manual.html
- ISO 8601: *Data elements and interchange formats -- Information interchange -- Representation of dates and times,* ISO 8601:2004, http://www.iso.org/iso/catalogue_detail?csnumber=40874
- JSON: *The application/json Media Type for JavaScript Object Notation (JSON),* http://tools.ietf.org/html/rfc4627
- IEEE LOM: *IEEE Standard for Learning Object Metadata*, IEEE Std 1484.12.1™-2002, IEEE Computer Society, September 2002.
- OAI-PMH: *The Open Archives Initiative Protocol for Metadata Harvesting*, V2.0, http://www.openarchives.org/OAI/openarchivesprotocol.html
- ▲OAUTH, http://oauth.net/
- RFC 4122: *A Universally Unique IDentifier (UUID) URN Namespace*, RFC 4122, http://www.ietf.org/rfc/rfc4122.txt
- ▲RFC 3880: *OpenPGP Messange Format,* http://tools.ietf.org/rfc/rfc4880.txt
- SRU *Search/Retrieval via URL Specifications*, SRU Version 1.2 Specifications, The Library of Congress, August 2007, http://www.loc.gov/standards/sru/specs/
- SWORD: *SWORD AtomPub Provide V 1.3*, http://www.swordapp.org/docs/sword-profile-1.3.html
- Unicode: *The Unicode Consortium. The Unicode Standard, Version 6.0.0*, http://www.unicode.org/versions/Unicode6.0.0/
- UTF-8: TBC (where in Unicode 6.0.0 doc?)

Change Log

NB: The change log only lists major updates to the specification.
NB: Updates may not results in a version update.

Version	**Date**	**Change**
0.10.0	20110117	Initial public release (lacks resource data model). Archived copy.
0.15.0	20110222	Added filtering. Added basic harvest. Added resource data model. Added

		basic delete. Archived copy.
0.16.0	20110310	Added OAI-PMH harvest. Obtain extended to support by doc_ID or by resource_ID/locator access. Documented Obtain. Basic Harvest extended to support doc_ID or by resource_ID/locator access for GetRecord. Archived copy.
0.17.0	20110415	Clarified that services are optional, described operational requirements to provision services. Revised XSD to support OAI-PMH harvest by resouce_ID. Clarified times are UTC 0. Clarified distributeAPI to uncouple it from other APIs. Add ToS checks to distribute and publish. Update reource doc to include ToS attribution. Added SWORD API. Archived copy.
0.20.0	201004xx	Added digital signature k-v pairs to resource documents. Added public key k-v pairs to all description documents. Updated network and node policy descripitons. Added signature, trust, ... verification to publish. Added identity section. Defined ALL behavior for obtain for size limited. Defined returning spec JSON vs. stored JSON for obtain and harvest. Authn (none, basic, SSH, Oauth, ...). Authz (access keys). Archived copy (link TBD)
Future	TBD	ToS attribution output to OAI. Flow control to OAI. Logging/tracking emit as paradata to services. Assertion (relation/sameas) and trust documents. Deduplication service. RESTful APIs.

Working Notes and Placeholder Text

How to submit/get attachments: publish, obtain, harvest get record