inBloom

Learning Map and inBloom Index System

Data Model and Server

inBloom Index Data Model and API Specification

V0.9 22 Jan 2013

Applied Minds, LLC

1209 Grand Central Ave Glendale CA 91201

Contents

Background

The Learning Map Data Model

The Bill & Melinda Gates Foundation (the Foundation) supports the implementation of the Common Core State Standards for US K-12 education. The Foundation awarded a contract to Applied Minds, LLC (AMI) to develop a Learning Map Data Model (LMDM) with the goal of it becoming a standard for educational technology infrastructure. The Learning Map will provide the organizing framework that maps the relationship between learning objectives, including dependencies and higher level groupings. It will also allow educational media resources, such as courses, books and web content to be linked to learning objectives. Curricula aligned to the standards will exhibit great diversity in highlighting paths through the Learning Map, and a suite of tools will allow authoring and visualization of the Learning Map. We believe that this data model will eventually enable the creation of online learning tools that are more responsive to the individual needs of a student. The LMDM is inspired by and based on the philosophy of the more general Knowledge Web (See: http://edge.org/conversation/aristotle-the-knowledge-web).

inBloom Technology

The Foundation has, in collaboration with the Carnegie Corporation, initiated an ambitious effort, Shared Learning Infrastructure (SLI), now inBloom Technology, to provide a new technology infrastructure that supports the Common Core Standards and the Foundation's vision, to be implemented by inBloom. AMI has been contracted by inBloom to build an implementation of a Learning Map and inBloom Index System, suitable for third-party software developers to populate content and develop applications.

inBloom Index Data Model and API Specification

Task 1 of the project requires AMI to develop a data model and API specification for the inBloom Index, formerly called the Learning Registry Index (LRI). This report presents the developer version (hence v0.9) of the data model. In addition to this report, we are delivering the following collections of schemata:

- **`bootstrap.json:`** A set of the minimal schema needed to operate the inBloom Index server.
- **`schema.org_fixed.json:`** An imported snapshot of schema.org schemata
- **`cc_schema.json:`** A snapshot of the Common Core State Standards schemata as defined in the Learning Map Data Model and API Specification delivered by AMI to the Gates Foundation on 15 Nov 2011.
- **`lrmi_fixed.json:`** An imported snapshot of lrmi.org schemata

Introduction

The inBloom Index is a searchable and browsable database of metadata sourced from data published by Learning Registry nodes. The inBloom Index is intended to provide a logically central service for finding and identifying structured data entities important to the inBloom Infrastructure. These entities include:

- Learning Objectives/Competencies, including educational standards such as the Common Core State Standards.
- Learning Resources, including instructional, assessment, and supplementary material.
- Complex metadata structures to be used for educational purposes, such as Competency Paths, statistical accumulations (e.g., Learning Resource paradata), ratings, and systems of annotation.

The inBloom Index will be hosted by an inBloom Index Server and accessible through a REST-like, HTTP-based API.

This document describes the current data model used by the inBloom Index and the API as presented by the inBloom Index Server.

Data Model Design Goals

The inBloom Index has two primary functions:

1. To be a broad-based repository of metadata for entities used in the inBloom Technology. This is the *archive* function of the inBloom Index.
2. To provide access mechanisms to the data in the repository that are sufficient for the required use cases. This is the *access* function of the inBloom Index.

The design of the inBloom Index data model includes technical features whose goal is to support one or both of these functions, including:

Archive Features

To service the Archive function well, the inBloom Index data model has been developed with the following design considerations.

- Only one logical data model exists. All variants are 1:1 mappable.
- The model is fully expressive for the requirements (no information is lost).
- The model is normalized (no duplication of data).
- The model is fully explicit (nothing implicit and undocumented).
- The model has an unambiguous representation of data.
- The model is self-compatible (no internal inconsistencies).

- The model is complete (no required elements are missing).

Access Features

To service the Access function well, the inBloom Index data model has been developed with the following design considerations.

- The model is easy to learn and use.
- The model is as simple as possible.
- The model is adapted to well known and accepted standards and technologies.
- The model is designed to serve a practical, useful API.

While the current inBloom Index data model does not achieve all of these design goals perfectly, they are the primary guidelines when making technical design choices.

inBloom Index Data Model

Overview

The inBloom Index data model is a mix of typed object design (as found in object oriented languages) and data graph model design (as found in many tuple/graph stores). Defining characteristics of this model include:

- It is composed of logical *entities* (distinct data structures) and their contained *properties* (literals or links to other entities).
- Real world objects (e.g., locations, organizations, documents) and abstractions (e.g., mailing address, event) are represented by inBloom Index entities.
- All entities have at least one (and preferably many) Globally Unique Identifiers (GUIDs).
- Entities must have one or more defining entity types:
  - Properties have exactly one property type.
  - Data literals have exactly one data type.
- Entity types exist in a super/subtype hierarchy.
- In general, properties on a particular entity may have a cardinality greater than one. (e.g., name aliases, required resources, members). Restriction to a singleton can be made in the specification of a property type.

Entity Types and Schema

Every entity instance in the inBloom Index must have one or more entity types. An entity type is primarily a non-exclusive categorization of an entity instance and has a specification of the properties that instances of that type are likely or expected to have.

To specify its properties, each entity type has a corresponding schema that enumerates and describes them. Every schema has the following properties itself, including:

- **urn:lri:property_type:comment:** A short description of the entity type
- **urn:lri:property_type:guid:** An opaque, ephemeral, internal GUID for this entity type *(unique/mandatory)*
- **urn:lri:property_type:id:** A pubic, permanent, external GUID for this entity type *(mandatory)*
- **urn:lri:property_type:name:** A human readable name
- **urn:lri:property_type:specific_properties:** A list of property type GUIDs that is likely to exist for instances specific to this type
- **urn:lri:property_type:supertypes:** Types whose schema are inherited by this type *(mandatory)*
- **urn:lri:property_type:url:** A canonical location on the Web where human readable information about this type may be found

Data Types

The inBloom Index data model supports a small number of literal value types. The data type hierarchy is as follows:

- **urn:lri:entity_type:data_type**
  - **urn:lri:data_type:Text** - A utf-8 encoded string.
    - **urn:lri:data_type:uri**
    - **urn:lri:data_type:url**
    - **urn:lri:data_type:datetime** - An ISO 8601 timestamp or duration
  - **urn:lri:data_type:number** - Abstract. Not used directly
    - **urn:lri:data_type:integer**
    - **urn:lri:data_type:float**
  - **urn:lri:data_type:boolean**

Property Types and Instances

A Property is the unit of data in the inBloom Index. It describes and specifies a single, specific assertion. Like entities, there are property types and instances of those types.

Property types have the following properties:

- **urn:lri:property_type:guid:** The internal GUID for this property type *(unique/mandatory)*
- **urn:lri:property_type:id:** An external GUID for this property type *(mandatory)*
- **urn:lri:property_type:name:** A human readable name
- **urn:lri:property_type:domains:** A set of entity types expected to have this property *(mandatory)*
- **urn:lri:property_type:ranges:** A set of entity types, of which one or more should be the value this property should have. Examples include: string (literal), person (link), postal address (link), gender (enumeration). *(mandatory)*
- **urn:lri:property_type:is_unique:** A boolean indicating whether more than one instance of this property should be allowed. The default is false.
- **urn:lri:property_type:mandatory:** A boolean indicating whether an instance of this property should always exist for entities of the given entity type. The default is false.
- **urn:lri:property_type:reverse:** A property type that is declared to be the same property type as this one but in the other direction. That reversed property must exist in the inBloom Index schema to insure that this property type schema is valid. Domains and ranges should be compatible with their reverse property schema. When a property with a reverse is created/updated, the reverse is implictly created/updated. *(unique)*

Property instances have the following attributes:

- **urn:lri:property_type:creator:** An internal GUID of the writer of this instance *(unique/mandatory)*
- **urn:lri:property_type:from:** An internal GUID of the "domain" entity instance *(unique/mandatory)*
- **urn:lri:property_type:replaced_by:** The internal GUID of a newer version of this property instance *(unique)*
- **urn:lri:property_type:timestamp:** The time this instance was created *(unique/mandatory)*
- **urn:lri:property_type:to:** The internal GUID of the "range" entity instance (is null for literal values) *(unique)*

- **urn:lri:property_type:value:** A number or string. For entity type ranges, the value is a GUID *(unique/mandatory)*
- **urn:lri:property_type:proptype:** The GUID of this instance's type *(unique/mandatory)*
- **urn:lri:property_type:alive:** A Boolean to indicate if this instance is alive or not ("deleted"). *(unique)* The default value is true.

Property Value Range Preferences

In the creation of a new property type, the following entity types should be used in decreasing order of range preference:

- Link to one of an enumeration or namespace of entities
- Type-constrained link to an Entity
- Constrained literal (e.g., numeric or date range)
- Unconstrained literal

Note on schema.org

The entity schemata in the inBloom Index data model are specifically intended to be "upwardly compatible" with object schema as defined by the schema.org effort. Many of the characteristics of schema.org schema have been used, including:

- Labels of common entity type and property type names.
- The "domains" and "ranges" model of property types.
- The JSON variant of the schema syntax.

The inBloom Index data model is not 100% downward compatible with the schema.org model. Rather, the inBloom Index data model imposes a number of additional restrictions and design priorities. In general this means that inBloom Index schemata will be directly convertible to valid schema.org schemata, but not always the other way around. These additional restrictions/priorities include:

- References to GUIDs are fully explicit (schema.org has implicit GUID components and usage)
- Properties are normalized (schema.org encourages denormalization)
- Entity Types and Property Types are explicit instances of the appropriate meta-types (in schema.org, these are implicit and can be inconsistent)
- Properties are not extensible (schema.org allows property extensions)
- The inBloom Index may at some point require entity-type-specific ranges (property ranges in schema.org are not entity-type specific)

- Enumerations only have entity instances as members (schema.org allows literal enumerations)
- Capitalizations in naming conventions of entity/property types in GUIDs are not used. Underscores are used instead. These can be mechanically mapped back to schema.org compatible names.
- The inBloom Index data model avoids using elements that are known to be broken in the schema.org model (e.g., geo-data is severely and ambiguously denormalized in schema.org)
- Property naming rules concerning cardinality are not followed. Instead, cardinality constraints are structurally explicit in the property type schema.
- The schema.org model for datatypes is not used. However, the inBloom Index model is generally compatible with the schema.org model.

Globally Unique Identifiers (GUIDs)

Globally Unique Identifiers (GUIDs) are used extensively in the inBloom Index Data Model as the primary handle for objects in the inBloom Index. In general, a user of the inBloom Index must have a GUID for an object in order to manipulate it, i.e., implicit unambiguous identification of an object by search constraint for write operations is not supported. Instead, a GUID must be retrieved and used as the handle for write operations. In general, if a data structure can be manipulated, it will be an inBloom Index entity, and so will have a GUID.

Expected Characteristics of GUIDs (Globally Unique Identifiers)

GUIDs should be treated as if they have the following characteristics:

- All entities should have at least one GUID, and will probably have more.
- Two entities with the same GUID are always the same entity.
- Two entities with different GUIDs may still be the same entity or may be different entities.
- All GUIDs should exist in an explicit namespace.
- GUID uniqueness is enforced during property instance creation/update.
- GUIDs are given in URN syntax. The path of the URN gives the hierarchy of namespaces in which the GUID lives.
- Namespaces should be treated as (at least implicit) entities.

inBloom Index Internal GUIDs

In the inBloom Index, every entity instance and every property instance has exactly one "internal" GUID or **`GUID`**. Internal GUIDs have the following characteristics distinct from normal (external, public) GUIDs:

- Internal GUIDs are 32 hexadecimal digits, not URIs.
- Internal GUIDs cannot be deleted or changed.
- Internal GUIDS exist to facilitate "housekeeping" functions for use of the inBloom Index. These uses include:
  - Discriminating otherwise seemingly identical instances of properties or entities
  - Providing a handle for properties so that they may be updated or deleted
  - Providing a "fallback" handle to an entity or property if other GUIDs are damaged or lost
- Internal GUIDS should be treated as ephemeral. While they are guaranteed internally consistent during a single inBloom Index query/response cycle, they may change over time, and so should not be expected to be stable over medium/long time spans or a very large number of queries.

Type GUID Syntax

Entity, property, and literal types currently have (external) GUIDs of the following URN form:

```
urn://<URN NAMESPACE>:entity_type | property_type | data_type:<OPTIONAL
NAMESPACE PATH>:<VALUE>
```

Examples:

- Entity: `urn:lri:entity_type:thing`
- Property: `urn:lri:property_type:name`
- Literal: `urn:lri:data_type:text`

The inBloom Index API

The inBloom Index server provides an HTTP-based API. Queries to the API are composed as HTTP GET or POST requests (both request types are treated the same). Each query has the following components:

- **`Hostname:`** The hostname or IP address of the inBloom Index server
- **`Object Type:`** This is one of "`entity`", "`property`", or "`document`".
- **`Verb:`** This is one of "`search`", "`create`", or "`update`".
- **`Parameters:`** These specify the query as well as response options. Each parameter is given as a name/value pair where the value is a URL encoded JSON object. Currently, the API supports the following parameters:
  - **`q:`** This is the specified query. The syntax of the query is dependent on the object type and verb of the query.

- **opts:** This is set of options that can be used to format or constrain the query. Currently, the following options are supported:
  - **details:** This is a boolean indicating whether each property returned as a search result includes detailed administrative fields. The default is false.
  - **format:** This specifies the format of the response. Currently supported values are: "json", "xml", and "yaml". The default is "json".
  - **access_token:** This is used for authentication.
  - **on_behalf_of:** This is used for a delegated User ID.

General API Request/Query Syntax

Each API request *(query)* must have the following form:

```
GET http://HOSTNAME/OBJECT_TYPE/VERB?q={}&opts={}
POST http://HOSTNAME/OBJECT_TYPE/VERB In POST parameters: q={}&opts={}
```

General API Response Syntax

In an API request, the object type/verb value pair defines the "action" of a query. While each action requires a different request data structure and each has a different result data structure, the following wrapper is common to all responses (given in JSON encoding).

```
{
   "api_version": <API RESPONSE VERSION NUMBER>,
   "cached": <A BOOLEAN INDICATING WHETHER THIS IS A CACHED RESULT>
   "opts": { <QUERY OPTIONS> },
   "q": { <ACTION-SPECIFIC QUERY> },
   "response":  <ACTION-SPECIFIC RESPONSE>,
   "status": <ONE OF "normal", "error">,
   "timestamp": <AN ISO 8601 TIMESTAMP>
}
```

Action-Specific Query and Response Syntax

The following are templates for the `q` and `response` elements of an API query for each action.

Basic entity search: /entity/search

This query returns entities with their property values for a given set of constraints. The query is composed of a set of "property type GUID"/ "equality match value" pairs and zero or more of the following global constraints:

`creator: <ASSERTER ID | [LIST OF ASSERTER IDs]>` -- Constrains by who wrote the searched assertions

`cursor: <A PREVIOUSLY RETURNED CURSOR>` - To allow for paging

`limit: <MAXIMUM NUMBER OF HITS RETURNED>` - Use as page size

`shape: <TREE SHAPE TO BE EXTRACTED>`

`constrain_by: <TREE SHAPE TO BE USED AS FILTERING CONSTRAINT>`

The response to an entity search is a list of dictionaries, one per entity returned. Each dictionary has the following name/value pairs:

`props:` A set of property name/values for this entity
`cursor:` An opaque cursor string for paging of results or `false` if there are no more results.

Examples:

Query: A simple entity lookup by ID

```
http://mylriserver.com/entity/search?q={"urn:lri:property_type:id":"urn
:ccss:standard:CCSS.Math.Content.3.NBT.A.2"}
```

Response:


```
{
    "action": "entity/search",
    "api_version": "0.5",
    "cached": false,
    "cursor": false,
    "opts": {},
    "q": {
        "urn:lri:property_type:id":
"urn:ccss:standard:CCSS.Math.Content.3.NBT.A.2"
    },
    "query_seconds": 0.223602,
    "response": [
        {
            "props": {
                "urn:ccss:property_type:ccid":
"CCSS.Math.Content.3.NBT.A.2",
                "urn:lri:property_type:contained_by": [
                    "urn:ccss:cluster:CCSS.Math.Content.3.NBT.A",
                    "urn:ccss:domain:CCSS.Math.Content.3.NBT"
                ],
                "urn:lri:property_type:creator": "LRI_ADMIN_USER_1",
                "urn:lri:property_type:description": "Fluently add and
subtract within 1000 using strategies and algor    ithms based on place
value, properties of operations, and/or the relationship between
addition and subtraction.",
                "urn:lri:property_type:guid":
"402254bdec3b1ec4b11bb1c6d323abb6",
                "urn:lri:property_type:id":
"urn:ccss:standard:CCSS.Math.Content.3.NBT.A.2",
                "urn:lri:property_type:name": "Fluently add and...",
                "urn:lri:property_type:timestamp": "2013-01-
14T04:53:13.387596",
                "urn:lri:property_type:types": [
                    "urn:ccss:entity_type:standard",
                    "urn:lri:entity_type:competency",
                    "urn:lri:entity_type:competency_container",
```

```
                    "urn:lri:entity_type:learning_objective",
                    "urn:lri:entity_type:thing"
                ]
            }
        }
    ],
    "status": "normal",
    "timestamp": "2013-01-14T23:13:34.368864"
}
```


Query: Get one of the entities that are standards in the math domain CCSS.Math.Content.3.NBT


```
http://mylriserver.com/entity/search?q={"urn:lri:property_type:containe
d_by":"urn:ccss:domain:CCSS.Math.Content.3.NBT","urn:lri:property_type:
types":"urn:ccss:entity_type:standard","limit":1}
```


Response:


```
{
    "action": "entity/search",
    "api_version": "0.5",
    "cached": false,
    "cursor": "eJyNT8FqxSAQ_JXgOYe4rhrzK-URXF0hEJJUTSE88u-
NLb33srPMDDPMW5STPk_OCxcxdR9vUfcHhQk62cFDMsYHFSw935g86ujkAAii78SR96NeBz
f_mbdpzcvUOM71mpswhX2rftk4znSJu-_-
0hGGIL3hZCGif7qcjqNLZFVwlBj_l95OEffrMZfqc_0ZgH3nwDZu5VT3L85z5nKuv6rAAUA
jRQ6KJAckKYlkMFGB8kSmNWuD0Wg23jKOBAoleGu0HgI_GFG87m944GDz",
    "opts": {},
    "q": {
        "limit": 1,
        "urn:lri:property_type:contained_by":
"urn:ccss:domain:CCSS.Math.Content.3.NBT",
        "urn:lri:property_type:types": "urn:ccss:entity_type:standard"
    },
    "query_seconds": 1.280359,
    "response": [
        {
            "props": {
                "urn:ccss:property_type:ccid":
"CCSS.Math.Content.3.NBT.A.3",
                "urn:lri:property_type:contained_by": [
                    "urn:ccss:cluster:CCSS.Math.Content.3.NBT.A",
                    "urn:ccss:domain:CCSS.Math.Content.3.NBT"
                ],
                "urn:lri:property_type:creator": "LRI_ADMIN_USER_1",
                "urn:lri:property_type:description": "Multiply one-
digit whole numbers by multiples of 10 in the range 10-90 (e.g., 9 X 80,
5 X 60) using strategies based on place value and properties of
operations.",
                "urn:lri:property_type:guid":
"349cd58cb79bc7b536c6ed7ff91b79f5",
```

```
                "urn:lri:property_type:id":
"urn:ccss:standard:CCSS.Math.Content.3.NBT.A.3",
                "urn:lri:property_type:name": "Multiply one-digit
whole...",
                "urn:lri:property_type:timestamp": "2013-01-
14T04:53:13.703244",
                "urn:lri:property_type:types": [
                    "urn:ccss:entity_type:standard",
                    "urn:lri:entity_type:competency",
                    "urn:lri:entity_type:competency_container",
                    "urn:lri:entity_type:learning_objective",
                    "urn:lri:entity_type:thing"
                ]
            }
        }
    ],
    "status": "normal",
    "timestamp": "2013-01-15T00:00:40.503547"
}
```


Query: Get the next hit from the previous query


```
http://mylriserver.com/entity/search?q={"cursor":"eJyNT8FqxSAQ_JXgOYe4r
hrzK-URXF0hEJJUTSE88u-
NLb33srPMDDPMW5STPk_OCxcxdR9vUfcHhQk62cFDMsYHFSw935g86ujkAAii78SR96NeBz
f_mbdpzcvUOM71mpswhX2rftk4znSJu-_-
0hGGIL3hZCGif7qcjqNLZFVwlBj_l95OEffrMZfqc_0ZgH3nwDZu5VT3L85z5nKuv6rAAUA
jRQ6KJAckKYlkMFGB8kSmNWuD0Wg23jKOBAoleGu0HgI_GFG87m944GDz","limit":1}
```


Response:


```
{
    "action": "entity/search",
    "api_version": "0.5",
    "cached": false,
    "cursor": "eJyNT0GKhDAQ_Ir02YPGTmL8yjBIJ-
mAIOokcUEG_77JLnOfS1VRVVTTb0infZ0cF04wNY835L0wKCeD7kgEpcgNTtuixkAovek7g
QLaBo64H_k6uPbPuE1rXKbqcczXXIPJ7VumZWM_2wvutvmso-
hcT4qDFh6p3DLSjyZYPThjA-N36xUS3M9STpli_nsA28YIXb2VQ95_OM6R07n-
pyAVeiVZkWYcrRiwF6SVlJ3jwh7hef8CeI9XOQ==",
    "opts": {},
    "q": {
        "cursor": "eJyNT8FqxSAQ_JXgOYe4rhrzK-URXF0hEJJUTSE88u-
NLb33srPMDDPMW5STPk_OCxcxdR9vUfcHhQk62cFDMsYHFSw935g86ukAAii78SR96NeBzf
_mbdpzcvUOM71mpswhX2rftk4znSJu-_-
0hGGIL3hZCGif7qcjqNLZFVwlBj_l95OEffrMZfqc_0ZgH3nwDZu5VT3L85z5nKuv6rAAUA
jR6KJAckKYlkMFGB8kSmNWuD0Wg23jKOBAoleGu0HgI_GFG87m944GDz",
        "limit": 1
    },
    "query_seconds": 0.047308,
    "response": [
        {
            "props": {
```

```
                "urn:ccss:property_type:ccid":
"CCSS.Math.Content.3.NBT.A.2",
                "urn:lri:property_type:contained_by": [
                    "urn:ccss:cluster:CCSS.Math.Content.3.NBT.A",
                    "urn:ccss:domain:CCSS.Math.Content.3.NBT"
                ],
                "urn:lri:property_type:creator": "LRI_ADMIN_USER_1",
                "urn:lri:property_type:description": "Fluently add and
subtract within 1000 using strategies and algorithms based on place
value, properties of operations, and/or the relationship between
addition and subtraction.",
                "urn:lri:property_type:guid":
"402254bdec3b1ec4b11bb1c6d323abb6",
                "urn:lri:property_type:id":
"urn:ccss:standard:CCSS.Math.Content.3.NBT.A.2",
                "urn:lri:property_type:name": "Fluently add and...",
                "urn:lri:property_type:timestamp": "2013-01-
14T04:53:13.387596",
                "urn:lri:property_type:types": [
                    "urn:ccss:entity_type:standard",
                    "urn:lri:entity_type:competency",
                    "urn:lri:entity_type:competency_container",
                    "urn:lri:entity_type:learning_objective",
                    "urn:lri:entity_type:thing"
                ]
            }
        }
    ],
    "status": "normal",
    "timestamp": "2013-01-15T00:06:07.096806"
}
```


Entity Creation: `/entity/create`

This query creates a new entity (with an automatically assigned new internal GUID). An initial set of properties are created at the same time. At least one explicit, external ID, and one type must be included. The properties are defined in the same format as entity search, except that the property values are being declared, not acting as constraints. Declared Values/IDs for properties may be singular or arrays (to declare multiple properties of the same type). The response is an entity record of the same format as in entity search results.

Query: Create an entity with four properties


```
http://mylriserver.com/entity/create?q={"urn:lri:property type:id":"urn
:unofficial-urn-
namespace:my_local_namespace:00001","urn:lri:property_type:types":["urn
:lri:entity_type:thing","urn:lri:entity_type:learning_resource"],"urn:l
ri:property_type:name":"Multiplication Video For Left Handed
People"}&opts={"access_token":"MY_ACCESS_TOKEN"}
```

Response:


```
{
    "action": "entity/create",
    "api_version": "0.5",
    "authenticated_creator_id": "LRI_ADMIN_USER_1",
    "opts": {
        "access_token": "MY_ACCESS_TOKEN"
    },
    "q": {
        "urn:lri:property_type:id": "urn:unofficial-urn-
namespace:my_local_namespace:00001",
        "urn:lri:property_type:name": "Multiplication Video For Left
Handed People",
        "urn:lri:property_type:types": [
            "urn:lri:entity_type:thing",
            "urn:lri:entity_type:learning_resource"
        ]
    },
    "query_seconds": 0.51587,
    "response": {
        "urn:lri:property_type:creator": "LRI_ADMIN_USER_1",
        "urn:lri:property type:guid":
"ee3138074837159fbb2029b4044b648a",
        "urn:lri:property_type:id": "urn:unofficial-urn-
namespace:my_local_namespace:00001",
        "urn:lri:property_type:name": "Multiplication Video For Left
Handed People",
        "urn:lri:property_type:timestamp":  "2013-01-15T22:44:38.551313",
        "urn:lri:property_type:types": [
            "urn:lri:entity_type:thing",
            "urn:lri:entity_type:learning_resource"
        ]
    },
    "status": "normal",
    "timestamp": "2013-01-15T22:44:38.550520"
}
```


Property Creation: `/property/create`

This action creates a new property value for an existing entity. The query has two name/value pairs:

- from: <INTERNAL GUID OF THE ENTITY> (urn:lri:property_type:guid property)
- <PROPERTY TYPE GUID>: <PROPERTY VALUE>

The response is the complete record for the property value.

Query: Simple addition of new property to an existing entity

```
http://mylriserver.com/property/create?q={"from":"ee3138074837159fbb2029b4044b648a","proptype":"urn:lri:property_type:teaches","value":"urn:ccss:standard:CCSS.Math.Content.3.NBT.A.3"}&opts={"access_token":"MY_ACCESS_TOKEN"}
```

Response:


```
{
    "action": "property/create",
    "api_version": "0.5",
    "authenticated_creator_id": "LRI_ADMIN_USER_1",
    "opts": {
        "access_token": "MY_ACCESS_TOKEN"
    },
    "q": {
        "from": "ee3138074837159fbb2029b4044b648a",
        "proptype": "urn:lri:property_type:teaches",
        "value": "urn:ccss:standard:CCSS.Math.Content.3.NBT.A.3"
    },
    "query_seconds": 0.023609,
    "response": {
        "alive": true,
        "complete": true,
        "creator": "LRI_ADMIN_USER_1",
        "from": "ee3138074837159fbb2029b4044b648a",
        "guid": "19304a6cb83a812960036e4c1ba8859c",
        "proptype": "urn:lri:property_type:teaches",
        "replaced_by": "",
        "timestamp": "2013-01-16T07:16:02.955944",
        "to": "349cd58cb79bc7b536c6ed7ff91b79f5",
        "value": "urn:ccss:standard:CCSS.Math.Content.3.NBT.A.3"
    },
    "status": "normal",
    "timestamp": "2013-01-16T07:16:02.943869"
}
```


Updating a property: /property/update

This query allows a property value to be updated. The property instance may only be referenced by its IGUID. The update is specified by updating values for one of the following property fields:

- `alive`: One of "true" (accessible) or "false" (hidden -- treat as deleted).
- `value`: The property instance value (literal or ID)

Specifying `"alive":false` deletes the property. Specifying `true` to update a deleted property un-deletes it. Whenever a property is updated, internally, the LRI creates a new property instance and sets the `replaced_by` field value to the internal GUID of the replacement. Searches in the LRI normally will only return live properties. The response to

an updated query contains two property instance records: the "old" property instance and the "new" property instance.

The response is the complete record for the property value.

Query: Simple update of property value

```
http://mylriserver.com/property/update?q={"guid":"19304a6cb83a812960036
e4c1ba8859c","value":"urn:ccss:standard:CCSS.Math.Content.4.NBT.B.5"}&o
pts={"access_token":"MY_ACCESS_TOKEN"}
```

Response:


```
{
    "action": "property/update",
    "api_version": "0.5",
    "authenticated_creator_id": "LRI_ADMIN_USER_1",
    "opts": {
        "access_token": "MY_ACCESS_TOKEN"
    },
    "q": {
        "guid": "19304a6cb83a812960036e4c1ba8859c",
        "value": "urn:ccss:standard:CCSS.Math.Content.4.NBT.B.5"
    },
    "query_seconds": 0.061631,
    "response": {
        "new": {
            "alive": true,
            "complete": true,
            "creator": "LRI_ADMIN_USER_1",
            "from": "ee3138074837159fbb2029b4044b648a",
            "guid": "2e53f2ecc89ae5b19ac383f34790f2e7",
            "proptype": "urn:lri:property_type:teaches",
            "replaced_by": "",
            "timestamp": "2013-01-16T07:17:26.657499",
            "to": "dd963a24df5f3184f5f276a466808a61"
        },
        "old": {
            "alive": true,
            "complete": true,
            "creator": "LRI_ADMIN_USER_1",
            "from": "ee3138074837159fbb2029b4044b648a",
            "guid": "19304a6cb83a812960036e4c1ba8859c",
            "proptype": "urn:lri:property_type:teaches",
            "replaced_by": "2e53f2ecc89ae5b19ac383f34790f2e7",
            "timestamp": "2013-01-16T07:16:02.955944",
            "to": "349cd58cb79bc7b536c6ed7ff91b79f5"
        }
    },
    "status": "normal",
    "timestamp": "2013-01-16T07:17:26.633807"
```

```
}
```

Query: Property Delete

```
http://mylriserver.com/property/update?q={"guid":"2e53f2ecc89ae5b19ac38
3f34790f2e7","alive":false}&opts={"access_token":"MY_ACCESS_TOKEN"}
```

Response:


```
{
    "action": "property/update",
    "api_version": "0.5",
    "authenticated_creator_id": "LRI_ADMIN_USER_1",
    "opts": {
        "access_token": "MY_ACCESS_TOKEN"
    },
    "q": {
        "alive": false,
        "guid": "2e53f2ecc89ae5b19ac383f34790f2e7"
    },
    "query_seconds": 0.040458,
    "response": {
        "new": {
            "alive": false,
            "complete": true,
            "creator": "LRI_ADMIN_USER_1",
            "from": "ee3138074837159fbb2029b4044b648a",
            "guid": "3bc5a7fb20d68dd0c697a0f8bfb99b64",
            "proptype": "urn:lri:property_type:teaches",
            "replaced_by": "",
            "timestamp": "2013-01-16T07:17:46.221920",
            "to": "dd963a24df5f3184f5f276a466808a61"
        },
        "old": {
            "alive": true,
            "complete": true,
            "creator": "LRI_ADMIN_USER_1",
            "from": "ee3138074837159fbb2029b4044b648a",
            "guid": "2e53f2ecc89ae5b19ac383f34790f2e7",
            "proptype": "urn:lri:property_type:teaches",
            "replaced_by": "3bc5a7fb20d68dd0c697a0f8bfb99b64",
            "timestamp": "2013-01-16T07:17:26.657499",
            "to": "dd963a24df5f3184f5f276a466808a61"
        }
    },
    "status": "normal",
    "timestamp": "2013-01-16T07:17:46.205852"
}
```

Query: Property Undelete

```
http://mylriserver.com/property/update?q={"guid":"3bc5a7fb20d68dd0c697a
0f8bfb99b64","alive":true}&opts={"access_token":"MY_ACCESS_TOKEN"}
```

Response:


```
{
    "action": "property/update",
    "api_version": "0.5",
    "authenticated_creator_id": "LRI_ADMIN_USER_1",
    "opts": {
        "access_token": "MY_ACCESS_TOKEN"
    },
    "q": {
        "alive": true,
        "guid": "3bc5a7fb20d68dd0c697a0f8bfb99b64"
    },
    "query_seconds": 0.075346,
    "response": {
        "new": {
            "alive": true,
            "complete": true,
            "creator": "LRI_ADMIN_USER_1",
            "from": "ee3138074837159fbb2029b4044b648a",
            "guid": "2fe85e56f99be4a377e4756c09507f67",
            "proptype": "urn:lri:property_type:teaches",
            "replaced_by": "",
            "timestamp": "2013-01-16T07:18:19.389750",
            "to": "dd963a24df5f3184f5f276a466808a61"
        },
        "old": {
            "alive": false,
            "complete": true,
            "creator": "LRI_ADMIN_USER_1",
            "from": "ee3138074837159fbb2029b4044b648a",
            "guid": "3bc5a7fb20d68dd0c697a0f8bfb99b64",
            "proptype": "urn:lri:property_type:teaches",
            "replaced_by": "2fe85e56f99be4a377e4756c09507f67",
            "timestamp": "2013-01-16T07:17:46.221920",
            "to": "dd963a24df5f3184f5f276a466808a61"
        }
    },
    "status": "normal",
    "timestamp": "2013-01-16T07:18:19.366909"
}
```


Shape Extraction Queries

The LRI server entity/search action can return more than one entity with its direct properties. Using the `shape` field in the query parameters, a tree of connected entities and

their properties can be included. The value for the shape of a query is a nested set of dictionaries, where values are either branches (contained dictionaries) or leaves (`null`). This extraction is considered to be "best effort" in the sense that partial branches are retrieved if they exist.

For example, the query:

Query: Entity search with shape tree extraction

```
http://mylriserver.com/entity/search?q={"urn:lri:property_type:id":"urn
:unofficial-urn-
namespace:my_local_namespace:00001","shape":{"urn:lri:property_type:tea
ches":{"urn:lri:property_type:contained_by":null}}}
```

Find which Competencies (e.g., CCSS standards) taught by Learning Resource with ID `urn:unofficial-urn-namespace:my_local_namespace:00001` and describe all of the containers (e.g., clusters and domains) that contain this standard.

This query would return a result tree, such as:

Response:

```
{
    "action": "entity/search",
    "api_version": "0.5",
    "cached": false,
    "cursor": false,
    "opts": {},
    "q": {
        "shape": {
            "urn:lri:property_type:teaches": {
                "urn:lri:property_type:contained_by": null
            }
        },
        "urn:lri:property_type:id": "urn:unofficial-urn-
namespace:my_local_namespace:00001"
    },
    "query_seconds": 0.221877,
    "response": [
        {
            "props": {
                "urn:lri:property_type:creator": "LRI_ADMIN_USER_1",
                "urn:lri:property_type:guid":
"ee3138074837159fbb2029b4044b648a",
                "urn:lri:property_type:id": "urn:unofficial-urn-
namespace:my_local_namespace:00001",
                "urn:lri:property_type:name": "Multiplication Video For
Left Handed People",
                "urn:lri:property_type:teaches": [
                    {
                        "props": {
```

```
                            "urn:ccss:property_type:ccid":
"CCSS.Math.Content.4.NBT.B.5",
                            "urn:lri:property_type:contained_by": [
                                {
                                    "props": {
                                        "urn:ccss:property_type:ccid":
"CCSS.Math.Content.4.NBT.B",

"urn:lri:property_type:contained_by":
"urn:ccss:domain:CCSS.Math.Content.4.NB    T",

"urn:lri:property_type:contains": [

"urn:ccss:standard:CCSS.Math.Content.4.NBT.B.4",

"urn:ccss:standard:CCSS.Math.Content.4.NBT.B.5",

"urn:ccss:standard:CCSS.Math.Content.4.NBT.B.6"
                                        ],
                                        "urn:lri:property_type:creator":
"LRI_ADMIN_USER_1",
                                        "urn:lri:property_type:guid":
"75a925b98a4351220fdc59b0bf3fd8a0",
                                        "urn:lri:property_type:id":
"urn:ccss:cluster:CCSS.Math.Content.4.NBT.B",
                                        "urn:lri:property_type:name":
"Math Cluster CCSS.Math.Content.4.NBT.B",

"urn:lri:property_type:timestamp": "2013-01-14T04:48:00.005247",
                                        "urn:lri:property type:types":
[

"urn:ccss:entity_type:cluster",

"urn:lri:entity_type:competency",

"urn:lri:entity_type:competency_container",

"urn:lri:entity_type:learning_objective",
                                            "urn:lri:entity_type:thing"
                                        ]
                                    }
                                },
                                {
                                    "props": {
                                        "urn:ccss:property_type:ccid":
"CCSS.Math.Content.4.NBT",

"urn:lri:property_type:contained_by":
"urn:ccss:grade:CCSS.Math.Content.4",

"urn:lri:property_type:contains": [
```

```
"urn:ccss:cluster:CCSS.Math.Content.4.NBT.A",

"urn:ccss:cluster:CCSS.Math.Content.4.NBT.B",

"urn:ccss:standard:CCSS.Math.Content.4.NBT.A.1",

"urn:ccss:standard:CCSS.Math.Content.4.NBT.A.2",

"urn:ccss:standard:CCSS.Math.Content.4.NBT.A.3",

"urn:ccss:standard:CCSS.Math.Content.4.NBT.B.4",

"urn:ccss:standard:CCSS.Math.Content.4.NBT.B.5",

"urn:ccss:standard:CCSS.Math.Content.4.NBT.B.6"
                                        ],
                                        "urn:lri:property_type:creator":
"LRI_ADMIN_USER_1",
                                        "urn:lri:property_type:guid":
"e403e0a125ca95f86e4d730c27058e94",
                                        "urn:lri:property_type:id":
"urn:ccss:domain:CCSS.Math.Content.4.NBT",
                                        "urn:lri:property_type:name":
"Math Domain CCSS.Math.Content.4.NBT",

"urn:lri:property_type:timestamp": "2013-01-14T04:48:44.873937",
                                        "urn:lri:property_type:types":
[

"urn:ccss:entity_type:domain",

"urn:lri:entity_type:competency",

"urn:lri:entity_type:competency_container",

"urn:lri:entity_type:learning_objective",
                                            "urn:lri:entity_type:thing"
                                        ]
                                    }
                                }
                            ],
                            "urn:lri:property_type:creator":
"LRI_ADMIN_USER_1",
                            "urn:lri:property_type:description":
"Multiply a whole number of up to four digits by a o    ne-digit whole
number, and multiply two two-digit numbers, using strategies based on
place value and the properties o    f operations. Illustrate and
explain the calculation by using equations, rectangular arrays, and/or
area models.",
                            "urn:lri:property_type:guid":
"dd963a24df5f3184f5f276a466808a61",
```

```
                            "urn:lri:property_type:id":
"urn:ccss:standard:CCSS.Math.Content.4.NBT.B.5",
                            "urn:lri:property_type:name": "Multiply a
whole...",
                            "urn:lri:property_type:taught_by":
"urn:unofficial-urn-namespace:my_local_namespace:00001",
                            "urn:lri:property_type:timestamp": "2013-
01-14T04:53:22.600560",
                            "urn:lri:property_type:types": [
                                "urn:ccss:entity_type:standard",
                                "urn:lri:entity_type:competency",

"urn:lri:entity_type:competency_container",

"urn:lri:entity_type:learning_objective",
                                "urn:lri:entity_type:thing"
                            ]
                        }
                    }
                ],
                "urn:lri:property_type:timestamp": "2013-01-
15T22:44:38.551313",
                "urn:lri:property_type:types": [
                    "urn:lri:entity_type:learning_resource",
                    "urn:lri:entity_type:thing"
                ]
            }
        }
    ],
    "status": "normal",
    "timestamp": "2013-01-16T23:51:42.745032"
 }
```


EXPERIMENTAL: Shape Constraints

The LRI provides for a limited ability to constrain an entity search by the IDs of other entities several hops away. To perform such a constraint, create a "constrain_by" clause with a tree of property types. This is similar to a shape extraction, except that the leaves of the tree must be an ID of an entity. ***NOTE:*** `constrain_by` is considered a constraint on other search parameters, and has very poor performance if used without reasonable other constraints.

Example: This query finds Learning Resources that teach Competencies (e.g.,standards) in the CSSS main domain `urn:ccss:domain:CCSS.Math.Content.4.NBT`.

Query: Entity search with shape constraint


```
http://mylriserver.com/entity/search?q={"urn:lri:property_type:types":"
urn:lri:entity_type:learning_resource","constrain_by":{"urn:lri:propert
y_type:teaches":{"urn:lri:property_type:contained_by":"urn:ccss:domain:
CCSS.Math.Content.4.NBT"}}}
```

Response:


```
{
    "action": "entity/search",
    "api_version": "0.5",
    "cached": false,
    "cursor": false,
    "opts": {},
    "q": {
        "constrain_by": {
            "urn:lri:property_type:teaches": {
                "urn:lri:property_type:contained_by":
"urn:ccss:domain:CCSS.Math.Content.4.NBT"
            }
        },
        "urn:lri:property_type:types":
"urn:lri:entity_type:learning_resource"
    },
    "query_seconds": 0.235874,
    "response": [
        {
            "props": {
                "urn:lri:property_type:creator": "LRI_ADMIN_USER_1",
                "urn:lri:property_type:guid":
"ee3138074837159fbb2029b4044b648a",
                "urn:lri:property_type:id": "urn:unofficial-urn-
namespace:my_local_namespace:00001",
                "urn:lri:property_type:name": "Multiplication Video For
Left Handed People",
                "urn:lri:property_type:teaches":
"urn:ccss:standard:CCSS.Math.Content.4.NBT.B.5",
                "urn:lri:property_type:timestamp": "2013-01-
15T22:44:38.551313",
                "urn:lri:property_type:types": [
                    "urn:lri:entity_type:learning_resource",
                    "urn:lri:entity_type:thing"
                ]
            }
        }
    ],
    "status": "normal",
    "timestamp": "2013-01-17T02:09:08.464761"
}
```


Detailed Property Instance Records

Normally, entity searches simply return the name/value pairs for the entities' properties. If there are identical property values for multiple property instances, then only one is returned.

If the complete property record is desired (e.g., in order to know the IGUID to update a property), then the `"details":true` option can be set in the query. In this case, all property instances are returned.

Example:

Query: Entity search with internals details of properties

```
http://mylriserver.com/entity/search?q={"urn:lri:property_type:id":"urn
:unofficial-urn-
namespace:my_local_namespace:00001"}&opts={"details":true}
```

Response:


```
{
    "action": "entity/search",
    "api_version": "0.5",
    "cached": false,
    "cursor": false,
    "opts": {
        "details": true
    },
    "q": {
        "urn:lri:property_type:id": "urn:unofficial-urn-
namespace:my_local_namespace:00001"
    },
    "query_seconds": 0.055838,
    "response": [
        {
            "props": {
                "urn:lri:property_type:creator": {
                    "LRI_ADMIN_USER_1": [
                        {
                            "alive": true,
                            "complete": true,
                            "creator": "LRI_ADMIN_USER_1",
                            "from": "ee3138074837159fbb2029b4044b648a",
                            "guid": "333b8907a1ea82b4f2afbf1eb7ebc594",
                            "proptype":  "urn:lri:property_type:creator",
                            "replaced_by": "",
                            "timestamp": "2013-01-15T22:44:38.919369",
                            "value": "LRI_ADMIN_USER_1"
                        }
                    ]
                },
                "urn:lri:property_type:guid": {
                    "ee3138074837159fbb2029b4044b648a": [
                        {
                            "alive": true,
                            "complete": true,
                            "creator": "LRI_ADMIN_USER_1",
```

```
                            "from": "ee3138074837159fbb2029b4044b648a",
                            "guid": "a610cf4078a10ba4353933e0502f0809",
                            "proptype": "urn:lri:property_type:guid",
                            "replaced_by": "",
                            "timestamp": "2013-01-15T22:44:38.878246",
                            "value": "ee3138074837159fbb2029b4044b648a"
                        }
                    ]
                },
                "urn:lri:property_type:id": {
                    "urn:unofficial-urn-
namespace:my_local_namespace:00001": [
                        {
                            "alive": true,
                            "complete": true,
                            "creator": "LRI_ADMIN_USER_1",
                            "from": "ee3138074837159fbb2029b4044b648a",
                            "guid": "0edb228d94c8cf137c2ced0b4bf61efb",
                            "proptype": "urn:lri:property_type:id",
                            "replaced_by": "",
                            "timestamp": "2013-01-15T22:44:38.825616",
                            "value": "urn:unofficial-urn-
namespace:my_local_namespace:00001"
                        }
                    ]
                },
                "urn:lri:property_type:name": {
                    "Multiplication Video For Left Handed People": [
                        {
                            "alive": true,
                            "complete": true,
                            "creator": "LRI_ADMIN_USER_1",
                            "from": "ee3138074837159fbb2029b4044b648a",
                            "guid": "616abe1e2591d4560de5cf714ab7ce64",
                            "proptype": "urn:lri:property_type:name",
                            "replaced_by": "",
                            "timestamp": "2013-01-15T22:44:38.855364",
                            "value": "Multiplication Video For Left
Handed People"
                        }
                    ]
                },
                "urn:lri:property_type:teaches": {
                    "urn:ccss:standard:CCSS.Math.Content.4.NBT.B.5": [
                        {
                            "alive": true,
                            "complete": true,
                            "creator": "LRI_ADMIN_USER_1",
                            "from": "ee3138074837159fbb2029b4044b648a",
```

```
                            "guid": "2fe85e56f99be4a377e4756c09507f67",
                            "proptype":  "urn:lri:property_type:teaches",
                            "replaced_by": "",
                            "timestamp": "2013-01-16T07:18:19.389750",
                            "to": "dd963a24df5f3184f5f276a466808a61"
                        }
                    ]
                },
                "urn:lri:property_type:timestamp": {
                    "2013-01-15T22:44:38.551313": [
                        {
                            "alive": true,
                            "complete": true,
                            "creator": "LRI_ADMIN_USER_1",
                            "from": "ee3138074837159fbb2029b4044b648a",
                            "guid": "90273d7ad501f53a83033aa92ba1a19f",
                            "proptype":
"urn:lri:property_type:timestamp",
                            "replaced_by": "",
                            "timestamp": "2013-01-15T22:44:38.900889",
                            "value": "2013-01-15T22:44:38.551313"
                        }
                    ]
                },
                "urn:lri:property_type:types": {
                    "urn:lri:entity_type:learning_resource": [
                        {
                            "alive": true,
                            "complete": true,
                            "creator": "LRI_ADMIN_USER_1",
                            "from": "ee3138074837159fbb2029b4044b648a",
                            "guid": "96416690131d4ffc845cdf3f90136dec",
                            "proptype": "urn:lri:property_type:types",
                            "replaced_by": "",
                            "timestamp": "2013-01-15T22:44:38.956578",
                            "to": "bc0af7fb5898d103e904e2e96f004895"
                        }
                    ],
                    "urn:lri:entity_type:thing": [
                        {
                            "alive": true,
                            "complete": true,
                            "creator": "LRI_ADMIN_USER_1",
                            "from": "ee3138074837159fbb2029b4044b648a",
                            "guid": "1fe8b64dfcd74092aab414c2f06a761f",
                            "proptype": "urn:lri:property_type:types",
                            "replaced_by": "",
                            "timestamp": "2013-01-15T22:44:38.936232",
```

```
                            "to": "7c8718de6e79859e4f7fc1bc5df4610d"
                        }
                    ]
                }
            }
        }
    ],
    "status": "normal",
    "timestamp": "2013-01-17T02:10:38.657404"
}
```


User IDs, Access Tokens, and Authentication

While the LRI has no ***read*** access control, it does have a simple set of ***write*** access controls, depending on the user.

Determining a User ID

The LRI must attach an authenticated User ID to each assertion written. There are three mechanisms for determining the User ID of a write (`entity/create`,`property/create`,`property/update`) action. Each of the three uses the `access_token` parameter to find the User ID:

1. A normal user would log in to any SLI application (at `slcedu.org`) using the OAuth mechanism provided. An authenticated login provides an access token that can be passed as a query option (e.g., `&opts={"access_token":"MY_SLI_ACCESS_TOKEN"}`). A normal user may perform any write except to schema entities and properties.
2. An administrative user would provide an access token that has been pre-determined in the LRI instance's `lri_config.json` file. This configuration file contains a map (`admin_access_tokens`) of administrative access token values to administrative User IDs. An administrative user may perform any write, including write to schema entities and properties.
3. The `lri_config.json` file may contain a `delegate_tokens` parameter that has a list of deletegate access tokens. If a delete access token is used, then the `on_behalf_of` parameter must be defined in `opts` to contain the literal User ID string. The user is considered to be a normal user without administrative privileges. This login mechanism is somewhat dangerous, and thus intended only for use of bulk imports from Learning Registry nodes.

Boolean Operators

The LRI search API provides for logical boolean operators in search constraints. There are currently three such mechanisms:

Logical AND with different property types

When more than one property value is specified in a search constraint, the result set is the intersectons of those constraints.

Logical AND with the same property type

Because the q={} query parameter must be valid JSON, repeated keys are not allowed in a query. In order to constrain by two different values for the same property type, a prefix of <X>~ may be placed in front of the property type ID, where <X> is valid single alphanumeric character. For example, the query:

```
http://mylriserver.com/entity/search?q={"1~urn:lri:property_type:types"
:"urn:lri:entity_type:learning_resource","2~urn:lri:property_type:types
":"urn:schema-org:entity_type:book"}
```

will only return entities that are both Learning Resources and books.

Logical OR *(EXPERIMENTAL)*

The LRI server supports logical OR in search queries, but if used carelessly, can result in very poor performance. To use logical OR, simply use an array of constraint values rather than a single value. For example:

```
http://mylriserver.com/entity/search?q={"urn:lri:property_type:types":[
"urn:lri:entity_type:learning_resource","urn:schema-
org:entity_type:book"]}
```

will return entities are either a Learning Resource or a book or both.

Logical NOT

The LRI server does not support logical NOT. In general in a graph database, logical NOT is not necessary to accomplish constraint of results and is usually not very performant. The occurrence of the need for NOT for more than syntactic convenience often indicates a suboptimal data model.