Learning Standards Digital Representation Specification: RESTful API

Granular Identifiers and Metadata for CCSS (GIM CCSS) Project

Daniel R Rehak

V 1.0

2013-09-12

Contents

Introduction

Note: This section is informative.

This document specifies the RESTful Application Programming Interface (API) that can be used to access or publish the digital representation of learning standards statements.

The API permits the Common Core State Standards (CCSS) statements developed in the *Granular Identifiers and Metadata for CCSS Project (GIM CCSS)* to be published and accessed through a Published Agent or other software application. API methods include:

- Retrieving the digital representation of learning standards statements (one or more) represented in either the JSON or XML encodings [GIM CCSS JSON, GIM CCSS XML].
- Publishing and updating the digital representation of learning standards statements using the JSON encoding [GIM CCSS JSON].

The API satisfies the requirements of the project scope [Scope] and of a Publishing Agent [Publishing Agent].

The target audience for this Specification is both users of the API and application developers implementing the API.

The document includes:

- **Notation** – Definitions of normative terms used in the Specification.
- **API Overview** – Use, core principles and features of the API design (informative).
- **API Common Characteristics** – Characteristics of all API messages:
    - **Resource URIs** – Representing URIs in messages.
    - **Resource-Related HTTP Headers** – Specifying message attributes through HTTP headers.
    - **Method Hosting** – Hosting the API methods.
    - **Resource Representation** – Representing learning standards statements as resources.
    - **Response and Error Handling** – Representing results and errors in responses.
    - **Authentication** – Including authentication data in messages.
    - **Versioning** – Representing API and resource versions in messages.
    - **Results Pagination** – Managing large response sets.
    - **Messaging** – Transporting messages.
    - **Extensions** – Possible API extensions.
    - **Resource Validation** – Validating learning standards statements before publication.
- **API Methods** – Description of each of the API methods.
    - **Standards Statement Retrieval** – Retrieving one or more learning standards statements.
    - **Standards Statement Metainformation Retrieval** – Metainformation for retrieving one or more learning standards statements.
    - **Identifier Retrieval** – Retrieving information about an identifier.
    - **Identifier Metainformation Retrieval** – Metainformation for retrieving information about an identifier.
    - **Standards Statement Publishing or Updating** – Publishing a new learning standards statement to the statement data store or updating an existing learning standards statement in the statement data store.
    - **Standards Statement Deletion** – Deleting a learning standards statement from the statement data store.
- **Implementation Considerations** – Additional considerations for a client using the API or a Publishing Agent that provides the API.
- **Security Considerations** – Security considerations for producing or consuming applications that use the API to publish or access learning standards statements.
- **Conformance** – Criteria for an application that implements the API to conform to this Specification.
- **Examples** – Sample API requests and responses (informative).
- **API Design Decisions** – Choices and rationale in designing the API (informative).

- **Glossary** – Definitions of terms used in this Specification.
- **Normative References** – Normative references to other specifications used in this Specification.
- **Informative References** – References to other documents (informative).
- **Annex: Resource URIs and Accept Headers** – Rules for validating and processing Resource URIs and Accept Headers.
- **Annex: Resource Publication** – Rules governing the publishing/update process.
- **Annex: Conformance Test Cases** – Summary of test cases used for conformance testing.
- **Annex: Browser Access URIs** – URIs in http scheme form for use in access through a web browser.
- **Annex: GIM CCSS Statement Taxon Structure** – Taxon Structure used to classify GIM CCSS statements (informative).

Notation

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

Unless otherwise noted, sections in this document are normative.

API Overview

Note: This section is informative.

The API is designed to provide the interface to a *Publishing Agent* as shown in Figure 1. An application uses the API to access an application server or other similar software that isolates the digital store of learning standards statements from external systems. The Application Server provides an HTTPD for external access, through the RESTful API, that hides the application server and statement data store behind the API facade. The API can be used in any similar computing environment, either to access statements held in the statement data store or to publish and manage statements in the statement data store.

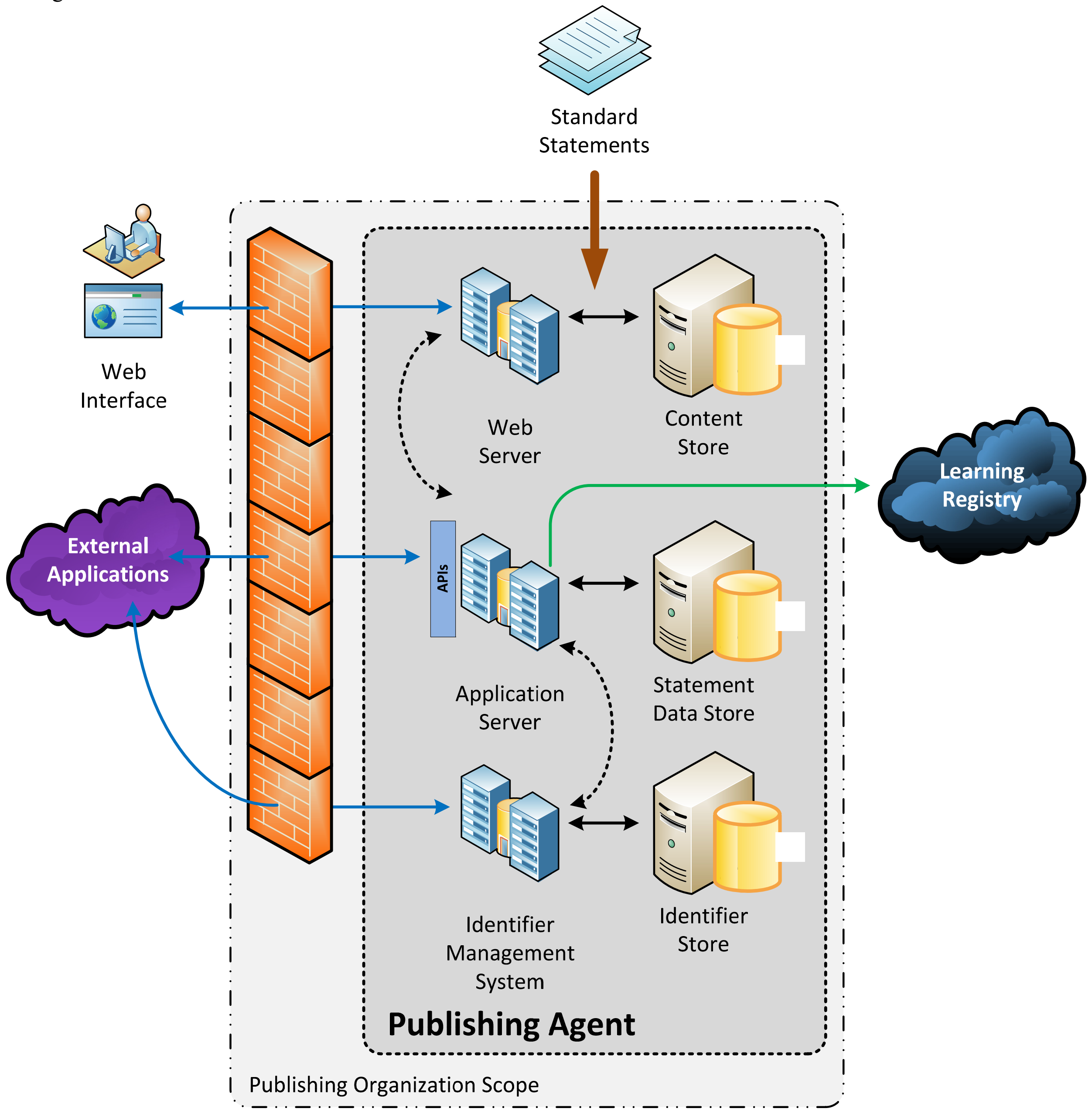

Figure 1: Publishing Agent Conceptual Design

API Principles

The API is based on the following design principles:

- *Abstract*: The API provides an abstraction layer that separates the Publishing Agent Statement Data Store from both external applications that produce and consume standards statements and from the Publishing Agent web component that provides a web interface to view and browse statements.
- *REST*: The API is designed as a fully RESTful API [REST].
- *Open*: API access methods are publically callable and are not access controlled. Learning standards statements can be retrieved through unsecured, unencrypted messages. Security and access controls are provided for publishing and management.
- *Minimal*: The API provides only essential methods to access and retrieve standards statements. Publishing statements and management are optional implementation features. Query of the statement data store is not included in the design.
- *Extensible*: All API methods and data models are extensible except as noted.
- *JSON Data Models*: JSON is the default and the preferred data serialization for standards statements. XML serialization can be obtained through the same methods.
- *Resource Oriented*: All standards statements and other data objects are assigned resolvable identifiers (URIs) and corresponding media types.

API Features

The API is RESTful. The API is defined in terms of HTTP methods, headers, results and resource names. HTTP features and URI resource names are used whenever possible.

An API call is an HTTP message, specifying the name of the standards statement as the resource request URI in the HTTP request. The message body contains the resource as the encoded representation of the standards statement. Specifics of a request are included in HTTP request headers. Additional response details are included in the HTTP response headers.

HTTP Methods: The HTTP methods supported by the API are:

- `GET` – retrieve a standards statement from the standards statement data store.
- `PUT` – store a new instance of a standards statement in the standards statement data store.
- `PUT` – update an existing standards statement in the standards statement data store (may create a new version).
- `DELETE` – delete a standards statement from the standards statement data store.
- `HEAD` – same as a `GET`, without the HTTP response body.

Resource Paths: The resource, i.e., the standards statement, is specified by:

- A unique opaque identifier for the statement, e.g., a UUID (`id/3f6eb648aa4e42ceae4f8e4022e933cf`).
- A path through the statement taxon classifiers (`statement/CCSS/Math/Content/./8/Math/G/A/1/a`).

The identifier form specifies a single standards statement. The path form can be used to specify a single standards statement or the collection of statements at a specific level in the taxon structure when the path ends with a slash (`/`).

Resource Representation: The JSON serialization of the standards statement is put in the body of the HTTP request when publishing (`PUT`) or updating (`PUT`) a statement in the standards statement data store. When requesting a resource (`GET`), the HTTP `Accept` header is used to specify the type of resource requested. An `Accept` header can be used to request either the JSON or XML serialization of the standards statement. If the path is for a collection, the Accept header can be used to request either the single statement at the root of the collection, or the collection of statements in the subtree starting at the specified root.

Response and Error Handling: All methods return an appropriate standard HTTP response code. When there is an error, a more detailed error message is returned as a JSON object in the body of the response.

Authentication and Encryption: Access requests do not require authentication. Access requests and responses (GET) may be transmitted using TLS encryption (HTTPS). Publish requests and responses (PUT) require authentication and TLS encryption. Authentication credentials are passed using an HTTP Authorization header. The API implementation selects the authentication method and specifies details of the Authorization header.

Versioning: API versions are specified through the request URI. Resources representations are versioned through the serialization format; each instance of a standards statement includes the version of the serialization used. API methods and resource representation can be updated independently, and each specific version of an API method will support some range of resource representation versions.

Results Pagination: When returning a large collection of standards statements, the response may only contain a portion of the collection. HTTP Link headers are used to indicate requests to obtain other portions of the collection.

API Common Characteristics

All of the API methods share common characteristics and design patterns using HTTP/1.1 [RFC 2616] and URIs [RFC 3986, RFC 3987]. Specifics of how these characteristics are used with each API method are included in the description of the method.

API Resource URIs

All HTTP requests SHALL include a resource URI composed of the following four parts. Each part SHALL be a URI path segment, separated by a slash (/).

- **API Base URI** – the API base URI SHALL be api
- **API Version** – as described below, the current API version SHALL be v1
- **Resource Type** – the type of resource:
  - For a learning standards statement, the resource type SHALL be statement
  - For an identifier, the resource type SHALL be id
  - For a vocabulary, the resource type SHALL be vocabulary
- **Resource Name** – the name of a specific resource or collection of resources.

A URI composed of the API Base URI, API Version and Resource Type followed by a slash (/) is REQUIRED in all messages. For a POST, PUT or DELETE message, the Resource Name is REQUIRED.

Note: The API does not support discovery of available API versions or resource types by omitting these parts of the URI and getting a response listing the available API versions or resources types. The API version and resource type must be present in all requests. The response structure to return a list of API versions or resource types has not been defined for this version of the API.

Learning Standards Statement Resource Names
The resource name for a learning standards statement is the order collection of statement taxon classifiers. Each is represented as an individual URI path segment. The *Statement Taxon Structure* Annex describes the classifiers used for GIM CCSS learning standards statements.

Resource Name Collections
The resource name may specify a single item or a collection of items of the same type.

- **Single Item** – If the Resource Name is not terminated with a slash (/), the URI SHALL be interpreted as specifying a single item, e.g., CCSS/Math/Content/./6/Math/NS/C/7 designates a single standards statement.

- **Collection of Items** – If the Resource Name terminates with a slash (/), the URI SHALL be interpreted as specifying the collection of items that are rooted in the resource hierarchy at the specified point, e.g., the resource name CCSS/Math/Content/./6/Math/NS/C/7/ designates the collection of all the “Components” for the “Standard”.

By default, the collection of items is only for the resources in the level in the hierarchy immediately below the specified item, not all items in the entire subtree below the root, e.g., CCSS/ specifies only the standards statements for the two CCSS initiatives, Math and ELA, not all CCSS Math and ELA statements. The collection also includes only the items below the specified root, not the root item itself.

Resource Name Parameters
The resource name MAY include one or more parameters as part of the path segment.

- **Recursive** – A path segment of the form ;r (semicolon followed by r – recursive traversal) added to the end of the resource name SHALL designate the contents of the entire subtree, e.g., CCSS;r specifies all CCSS statements. If the item specified is a single item, the item SHALL be included in the collection. If the item specified is a collection (ends in a slash (/), the root item SHALL NOT be included in the collection.

- **Size** – A path segment of the form ;size=dd (semicolon followed by size= and a non negative integer) added to the end of the resource name SHALL indicate that the user agent is requesting a maximum page size for a response. See Pagination below for additional details.

- **Locator** – A path segment of the form ;loc (semicolon followed by loc – locator) added to the end of a resource name that is an identifier SHALL indicate that the user agent is requesting only the *locator* for the identifier, i.e., to convert the identifier to a locator. The response body SHALL be empty; the locator shall be returned in the Content-Location response header.

- **Web Locator** – A path segment of the form ;http (semicolon followed by http – web-resolvable locator) added to the end of a resource name that is an identifier SHALL indicate that the user agent is requesting only the *locator* for the identifier, i.e., to convert the identifier to a web-resolvable locator, e.g., an HTTP URI. The response body SHALL be empty; the locator shall be returned in the Content-Location response header.

- **Version** – A path segment of the form ;v=vxx (semicolon followed by v= and a version label) added to the end of any taxon classifier path segment in a resource name for a statement SHALL indicate a specific version of the statement. See API Versioning below for additional details.

Note: The version parameter can be added to ANY path segment, the other parameters are added only to the END of the ENTIRE path. See the Design Decisions section for additional rationale.

The recursive and size parameters MAY be used together and MAY appear in any order. The locator parameter SHALL NOT be combined with any other parameter.

API Resource-Related HTTP Headers

An HTTP request MAY include one or more HTTP headers describing the resource.

An Accept header is used to specify the resource to be returned for a GET request.
- A JSON media type (e.g., application/vnd.ccss.standardstatement+JSON) SHALL be used to request a result in JSON.
- An XML media type (e.g., application/vnd.ccss.standardstatement+XML) SHALL be used to request a result in XML.

If the Accept header is omitted, the request SHALL be interpreted as a request for the JSON serialization.

If the resource URI specifies an item, the Accept header MAY be:
- Omitted – the request SHALL be interpreted as a request for a JSON serialization of the requested resource.
- application/vnd.ccss.standardstatement+JSON – the request is for the JSON serialization of a statement; the resource URI MUST specify a statement or an identifier.
- application/vnd.ccss.standardstatement+XML – the request is for the XML serialization of a statement; the resource URI MUST specify a statement or an identifier.
- application/vnd.ccss.vocabulary+JSON – the request is for the JSON serialization of a vocabulary; the resource URI MUST specify a vocabulary.
- Other – the request SHALL return an error; the request media type is incompatible with the resource URI.

If the resource URI specifies a collection, the Accept header MAY be:
- Omitted – the request SHALL be interpreted as a request for a JSON serialization of the requested collection.

- application/vnd.ccss.standardstatementcollection+JSON – the request is for the JSON serialization of a statement collection; the resource URI MUST specify a statement collection.
- application/vnd.ccss.standardstatementcollection+XML – the request is for the XML serialization of a statement collection; the resource URI MUST specify a statement collection
- Other – the request SHALL return an error; the request media type is incompatible with the resource URI.

The rules for processing different combinations of Resource URIs and Accept headers are presented in the *Resource URIs and Accept Headers* Annex.

Note: The interpretation of the different combinations of Resource Names, Resource Name Parameters and Accept headers is subject to review and revision based on feedback from the Technical Working Group.

All successful responses SHALL include a Content-Location header. The Content-Location header SHALL be a URI that can be used in a subsequent API message to specify the resource.

All responses that include a non-empty body SHALL include a Content-Type header. The Content-Type header SHALL be the media type returned in the body. Media types are defined below.

API Method Hosting

All HTTP requests SHALL include a HOST HTTP header specifying the target URI of the API method.

A deployment of the API SHALL establish the host URI (domain name of the Application server and port).

A deployment of the API SHOULD use the same host URI for all API methods.

API Resource Representation

Learning standards statements (individual statements or collections) and vocabularies SHALL be encoded either as JSON objects [RFC 4627] or in XML documents [XML]. Document media types SHALL conform to [RFC 4288] and SHOULD be registered [RFC 4298].

A JSON learningStandardsStatement object SHALL conform to the JSON serialization [GIM CCSS JSON]. The media type (e.g., in an Accept or Content-Type HTTP header) SHALL be application/vnd.ccss.standardstatement+JSON

A JSON learningStandardsStatementCollection object SHALL conform to the JSON serialization [GIM CCSS JSON]. The media type (e.g., in an Accept or Content-Type HTTP header) SHALL be application/vnd.ccss.standardstatementcollection+JSON

A JSON vocabulary object SHALL conform to the JSON serialization [GIM CCSS JSON]. The media type (e.g., in an Accept or Content-Type HTTP header) SHALL be application/vnd.ccss.vocabulary+JSON

An XML learningStandardsStatement document SHALL conform to the XML serialization [GIM CCSS XML]. The media type (e.g., in an Accept or Content-Type HTTP header) SHALL be application/vnd.ccss.standardstatement+XML

An XML learningStandardsStatementCollection object SHALL conform to the XML serialization [GIM CCSS XML]. The media type (e.g., in an Accept or Content-Type HTTP header) SHALL be application/vnd.ccss.standardstatementcollection+XML

An XML vocabulary object SHALL conform to the XML serialization [GIM CCSS XML]. The media type (e.g., in an Accept or Content-Type HTTP header) SHALL be application/vnd.ccss.vocabulary+XML

API Response and Error Handling

All API methods SHALL return a standard HTTP status code.

Successful Response
For a successful response, the API SHALL return a general success HTTP status code. The status code, response and Location response header for each method SHALL be:

- GET – Status Code [200, OK]. Returned when requesting a learning standards statement, collection or vocabulary. The requested resource SHALL be included in the response body.
- GET – Status Code [303, See Other]. Returned when requesting the locator for a resolvable identifier. The Location header SHALL be the URI that can be used to access the standards statement from the standards data store. The response body SHALL be empty.
- PUT – Status Code [201, Entity Created]. Returned when a new instance of the learning standards statement is created. The Location header SHALL be the URI that can be used to access the newly created instance of the standards statement from the standards statement data store. The response body SHALL be empty
- PUT – Status Code [201, Entity Created]. Returned when a new version of the learning standards statement is created. The Location header SHALL be the URI that can be used to access the newly created version of the standards statement from the standards data statement store. The response body SHALL be empty.
- PUT – Status Code [204, No Content]. Returned when a new version of the learning standards statement is not created (the existing version is updated in place). The Location header SHALL be the URI that can be used to access the standards statement from the standards statement data store. The response body SHALL be empty.
- DELETE – Status Code [204, No Content]. Returned when deleting a learning standards statement. The response body SHALL be empty.
- HEAD – Status Code [200, OK]. Returned when requesting metainformation for a learning standards statement, collection or vocabulary. The response body SHALL be empty.

Unsuccessful Response
If the request is valid and well formed, and the API Resource URI does not exist, the API SHALL return a “not found” HTTP status code. The status code and response for each method SHALL be:

- GET – Status Code [404, Not Found]. Returned when requesting a learning standards statement, collection or vocabulary that does not exist.
- PUT – Status Code [404, Not Found]. Returned when attempting to update a learning standards statement that does not exist.
- DELETE – Status Code [404, Not Found]. Returned when attempting to delete a learning standards statement that does not exist.
- HEAD – Status Code [404, Not Found]. Returned when requesting metainformation for a learning standards statement, collection or vocabulary that does not exist.

If the request is valid and well formed, and the API Resource URI does exist, the API SHALL return a “method not allowed” HTTP status code:

- PUT – Status Code [405, Method Not Allowed]. Returned when attempting to publish a learning standards statement that already exists.

API Errors
When there is an error, the API SHALL return the most specific applicable HTTP status code. In addition to the status code, a JSON object named "error" containing five required key-values pairs describing the error SHALL be returned in the body of the response:

- HTTP Status Code – integer – the HTTP status code.

- HTTP Status – string – the textual description of the status code.
- API Error Code – string – the API error code.
- API Error Description – string – the textual description of the error.
- The original HTTP request – string – the original HTTP request message.

The schema for the error object is represented using [JSON Schema Core, JSON Schema Validation].

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "version": "GIM-CCSS 20130212",
  "id": "http://publishingagent.tld/schema/error.json#",
  "schemaLicense": "Copyright © 2013, State Educational Technology Directors Association (SETDA).  This
work is made available under the terms of the Creative Commons Attribution 3.0 Unported (CC BY 3.0)
http://creativecommons.org/licenses/by/3.0/",
  "title": "API Error ",
  "description": "Schema for an API error description.",
  "type": "object",
  "required": [
    "error"
  ],
  "additionalProperties": false,
  "properties": {
    "error": {
      "id": "#error",
      "title": "API Error",
      "description": "API Error data.",
      "type": "object",
      "required": [
        "httpStatusCode",
        "httpStatus",
        "apiErrorCode",
        "apiErrorDescription",
        "apiRequest"
      ],
      "additionalProperties": false,
      "properties": {
        "httpStatusCode": {
          "id": "#httpStatusCode",
          "title": "http Status Code",
          "description": "The numeric http status code, 404.",
          "type": "integer"
        },
        "httpStatus": {
          "id": "#httpStatus",
          "title": "http Status",
          "description": "The http status name, e.g., NOT FOUND.",
          "type": "string"
        },
        "apiErrorCode": {
          "id": "#apiErrorCode",
          "title": "API Error Code",
          "description": "API Error Code.",
          "type": "string"
```

```
            },
            "apiErrorDescription": {
               "id": "#apiErrorDescription",
               "title": "API Error Description",
               "description": "Natural language description of the error.",
               "type": "string"
            },
            "apiRequest": {
               "id": "#apiRequest",
               "title": "API Request",
               "description": "Original API request – HTTP message.",
               "type": "string"
            }
         }
      }
   }
}
```

Note: JSON schema last validated: 20130217. JSON last validated: 20130217.

To Do: Assign URI for hosting the schema.

The schema SHALL NOT be extended.

The version number for this version of the schema is: GIM-CCSS 20130212.

Any significant change to the schema SHALL be indicated by a new schema version number.

The Content-Type header for an error response SHALL be the MIME media type for the error object. The MIME media type for the error object SHALL be:

- Type name: application
- Subtype name: JSON

Note: The error description text is not internationalized.

Errors applicable to all APIs and conditions are listed in the tables below. Errors for specific cases, e.g., authentication, resource validation, are described separately.

The API Error Description is not normative. An implementation MAY provide more descriptive and detailed error descriptions.

General Request Errors

API Error Code	API Error Description (Not normative)	HTTP Status Code	HTTP Status
Request-0000	Bad Request (more specific information not available)	400	Bad Request
Request-0001	Bad Resource URI (more specific information not available)	400	Bad Request
Request-0002	Resource base URI omitted	400	Bad Request
Request-0003	Resource base URI not valid (not api)	400	Bad Request
Request-0004	API Version omitted	400	Bad Request
Request-0005	API Version not supported	501	Not Implemented
Request-0006	Resource Type omitted	400	Bad Request
Request-0007	Resource Type not valid	400	Bad Request

API Error Code	API Error Description (Not normative)	HTTP Status Code	HTTP Status
Request-0008	Resource Name omitted	400	Bad Request
Request-0009	Resource Name not valid	400	Bad Request
Request-0010	Resource Name is a collection when only a statement is allowed	400	Bad Request
Request-0101	Media type is for a collection when resource name is for an item	406	Not Acceptable
Request-0102	Media type is for an item when resource name is for a collection	406	Not Acceptable
Request-0103	Media type is not a statement media type	406	Not Acceptable
Request-0104	Media type is not a vocabulary media type	406	Not Acceptable
Request-0105	A recursive subtree request is not applicable to a single item	406	Not Acceptable
Request-0106	A pagination request is not applicable to a single item	406	Not Acceptable
Request-0107	A pagination request is not applicable to the resource	406	Not Acceptable
Request-0108	A version request is not valid	400	Bad Request
Request-0109	A version request is not applicable to the resource	406	Not Acceptable
Request-0110	A version cannot be used with POST if a version already exists	400	Bad Request
Request-0201	GET/HEAD/DELETE includes body	400	Bad Request
Request-0202	POST/PUT data missing	400	Bad Request
Request-0203	Resource cannot be DELETED	405	Method Not Allowed
Request-0204	Attempting to publish a learning standards statement that already exists	405	Method Not Allowed

Service Errors

API Error Code	API Error Description (Not normative)	HTTP Status Code	HTTP Status
SERVICE-0000	Service Error (more specific information not available)	500	Server Error
SERVICE-0001	Feature not implemented (more specific information not available)	501	Not Implemented
SERVICE-0002	POST (publishing) not implemented	501	Not Implemented
SERVICE-0003	PUT (updating) not implemented	501	Not Implemented
SERVICE-0004	DELETE (deleting) not implemented	501	Not Implemented
SERVICE-0005	HEAD not implemented	501	Not Implemented
SERVICE-0006	Paginated Results not implemented	501	Not Implemented

API-Specific Errors

API Error Code	API Error Description (Not normative)	HTTP Status Code	HTTP Status
{API}-0000	API Error (more specific information not available)	500	
{API}-****	Specific API error description	500	

API-specific errors incorporate a 4-character code.
The first character denotes the type of error:

- 0 – Not specified.
    - 1 – Access and authentication errors.
    - 2 – Bad requests.
    - 3 – API operational errors.
    - 9 – Other unclassified API errors.
- The second character denotes the API group:
    - 0 – Applies to all APIs.
    - 1 – GET errors.
    - 2 – POST errors.

  - 3 – PUT errors.
  - 4 – DELETE errors.
  - 5 – HEAD errors.
- The last two characters are error-specific indicators. The same characters are used for the same error in different APIs.

API Authentication

Specific API methods may require authentication. Authentication credentials SHALL be included in an Authorization HTTP request header.

HTTP messages that include an Authorization header SHALL be encrypted using TLS.

A deployment of the API SHALL specify the details of the Authorization header value and the authentication process. A deployment of the API SHALL use the same authentication process for all API methods.

An API method that requires authentication and that does not include an Authorization header in the API request SHALL return an Auth-0001 error response [401, Unauthorized].

An API method that fails authentication SHALL return an Auth-0002 error response [403, Forbidden].

A deployment of the API method SHALL specify additional specific API authorization error responses. The authorization error responses SHALL NOT enable user agent discovery of behaviors that would compromise security, e.g., specific details of authorization errors SHOULD NOT be included in responses.

When a more specific API authorization error response is not available, the API SHALL return an Auth-0000 error response [403, Forbidden].

Authentication and Authorization Errors

API Error Code	API Error Description (Not normative)	HTTP Status Code	HTTP Status
Auth-0000	Authentication/Authorization error (not specific)	403	Forbidden
Auth-0001	Authentication required	401	Unauthorized
Auth-0002	Authentication failed	403	Forbidden

API Versioning

The API version SHALL be included as the second path element in the HTTP request URI.

The API version for this version of the Specification SHALL be v1

Modification of an API method, e.g., changing the behavior of the API method or adding features to the method, SHALL result in a new version of the API.

Each API method SHALL specify the versions of the resources schemata (standards statements and collections schemata) that the API version supports. The resource schema version SHALL be encoded in the resource representation. The resource version SHALL NOT be represented in the media type.

A Resource Name with a path segment of the form ;v=vxxx (version: semicolon followed by v= and a version label) added to the end of any path segment in the classifier in a resource name for a statement SHALL indicate a specific version of the standards statement. If the version parameter is malformed, the API SHALL return the Request-0003 error response [400, Bad Request]. Versioning schemes for standards statement are described in [Best Practices].

API Results Pagination

When returning a learning standards statement collection, the API method MAY return either the entire collection in one response, or it MAY return a part of the collection and require the user agent make repeated requests to retrieve the rest of the collection. The API implementation SHALL decide if the entire response is returned, or if the results are returned in groups/pages through pagination.

If the API method returns the entire collection in one response, the response SHALL NOT include a Link response header.

If the API method returns a part of the collection through pagination, the response SHALL include one or two Link response headers [RFC 5988]:

- The API Resource URI path to obtain the next page of results in the collection SHALL be specified in a Link response header with parameter rel="next". This header SHALL NOT be included when the results include the last part of the collection.
- The API Resource URI path to obtain the previous page of results in the collection SHALL be specified in a Link response header with parameter rel="prev". This header SHALL NOT be included when the results are the first part of the collection.

To page through the results collection, the user agent SHALL send a GET message with the API Resource URI from the Link header from the previous response to request another part/page of the collection. The API implementation shall determine the form of the URI: it need not be the same format as other API requests; it MAY include state information. The API implementation SHALL NOT use other HTTP features (e.g., Cookies) to support state.

The API implementation SHALL maintain the (view of the) collection and the Link URIs for sufficient time (determined by the API implementation) to allow the requesting client to obtain all the results. Pagination URIs MAY NOT be valid beyond a certain time determined by the API implementation.

During access of the collection, other asynchronous operations (e.g., publishing or deleting a standards statement) MAY invalidate the collection. The API implementation MAY immediately invalidate the pagination set, or it MAY hold a view for sufficient time for the requesting user agent to obtain all results. The API implementation SHALL determine the reset behavior.

To invalidate the pagination set, the API SHALL return a Pagination-0001 error response [409, Conflict], without any response body or Link headers as a response to a request including a URI for a part of the collection. The response SHALL include a Location header with an API Resource URI that can restart the request for the entire collection. The user agent needs to restart the retrieval with the new API Resource URI or the original request.

After pagination reset, all prior pagination URIs SHALL be invalid. Requesting an invalid pagination URI SHALL return a general “unsuccessful” error response [404, Not Found].

When returning pages of results, the total statements object of the collection SHALL specify the number of results in the returned page, not the total number of results in the entire collection.

A path segment of the form ;size=dd (semicolon followed by size= and a non negative integer) added to the end of the resource name SHALL indicate that the user agent is requesting a maximum number of statements for a response. If the API implementation does not support user-requested page sizes, the API SHALL return the Pagination-0003 error response [501, Not Implemented]. If the request size is invalid, the API SHALL return the Pagination-0002 error response [400, Bad Request].

If the API implementation supports user-requested page sizes, the API method SHALL return the collection in parts that contain no more that the requested number of statements.

Pagination Errors

API Error Code	API Error Description	HTTP Status Code	HTTP Status
Pagination-0000	Pagination error (more specific information not available)	500	Server Error
Pagination-0001	Pagination reset	409	Conflict
Pagination-0002	Specified page size is invalid (not a valid integer)	400	Bad Request
Pagination-0003	Client page size not supported	501	Not Implemented

API Messaging

All API methods SHALL use HTTP.

All HTTP messages SHALL use HTTP/1.1.

All HTTP requests and responses that include authentication or authorization credentials or that transport private or confidential data SHALL use TLS (Transport Layer Security) to encrypt messages [RFC 5246].

All API methods that do not require authentication or authorization credentials or that transport private or confidential data SHOULD NOT use TLS.

The HTTP POST message is reserved for use in a subsequent version of this Specification.

API Extensions

API extensions are permitted except as noted.

API Resource Validation

A learning standards statement submitted for publication in the statement data store shall be validated before being stored. An API method SHALL perform appropriate validation. Validation SHALL include:

- **Well Formed** – Validation that the form of the object or document is well formed, e.g., valid JSON syntax.
- **Schema Validation** – Validation that the object or document conforms to the schema specified in the document.
- **Semantic Validation** – Validation that the object or document conforms to all specified semantic constraints.

An API method SHALL fail and SHALL return the most detailed appropriate error response if the learning standards statement is not valid.

Resource Validation Errors

API Error Code	API Error Description (Not normative)	HTTP Status Code	HTTP Status
Validation-0000	Resource validation error (more specific information not available)	400	Bad Request
Validation-0101	JSON object not well formed (not valid JSON)	400	Bad Request
Validation-0102	JSON object does not contain a schema object	400	Bad Request
Validation-0103	JSON object schema object is invalid	400	Bad Request
Validation-0104	JSON object does not validate against the specified schema	400	Bad Request
Validation-0201	XML document not well formed (not valid XML)	400	Bad Request
Validation-0202	XML document does not contain a schema object	400	Bad Request
Validation-0203	XML document schema object is invalid	400	Bad Request
Validation-0204	XML document does not validate against the specified schema	400	Bad Request

API Error Code	API Error Description (Not normative)	HTTP Status Code	HTTP Status
Validation-1000	Learning standards statement semantic error (more specific information not available)	400	Bad Request
Validation-1101	Learning standards statement version invalid	400	Bad Request
Validation-1200	Identifier error (more specific information not available)	400	Bad Request
Validation-1211	Identifier type is not valid	400	Bad Request
Validation-1212	Identifier format is not valid for the specified identifier type	400	Bad Request
Validation-1300	Classification error (more specific information not available)	400	Bad Request
Validation-1311	Taxon vocabulary is not valid	400	Bad Request
Validation-1312	Taxon value is not valid for the specified taxon vocabulary	400	Bad Request
Validation-1313	Taxon value does not correspond to Resource Name	400	Bad Request
Validation-1321	Statement type vocabulary is not valid	400	Bad Request
Validation-1322	Statement type value is not valid for the specified statement type vocabulary	400	Bad Request
Validation-1390	Classification is not valid	400	Bad Request
Validation-1400	Educational characteristic error (more specific information not available)	400	Bad Request
Validation-1401	Grade level vocabulary is not valid	400	Bad Request
Validation-1402	Grade level value is not valid for the specified grade level vocabulary	400	Bad Request
Validation-1490	Educational characteristic is not valid	400	Bad Request
Validation-1500	Text error (more specific information not available)	400	Bad Request
Validation-1510	Text encoding vocabulary is not valid	400	Bad Request
Validation-1511	Text encoding value is not valid for the specified text encoding vocabulary	400	Bad Request
Validation-1521	Text language vocabulary is not valid	400	Bad Request
Validation-1522	Text language value is not valid for the specified text language encoding	400	Bad Request
Validation-1531	Text encoding is not valid for the specified text encoding	400	Bad Request
Validation-1600	Relationship error (more specific information not available)	400	Bad Request
Validation-1611	Relation type vocabulary is not valid	400	Bad Request
Validation-1612	Relation type value is not valid for the specified relation type vocabulary	400	Bad Request
Validation-1700	Metadata error (more specific information not available)	400	Bad Request
Validation-1701	Metadata attribute error	400	Bad Request
Validation-1702	Metadata value is not value for specified metadata attribute	400	Bad Request

Additional validation errors MAY be defined.

API Method: Standards Statement Retrieval

API Feature	Value	Notes
Summary	Retrieve learning standards statements from the statement data store for: • An individual statement or version of a statement. • The collection of all statements at a given level in the statement classification tree. • The entire subtree of statements below a given level in the statement classification tree.	
HTTP Method	GET	
HTTP Protocol	HTTP/1.1	
API Base URI	api/	
Version	v1/	
Resource Type	statement/	statement/ is used to designate a statement or statement collection.
	page/	page/ is used to designate a page of a results. Used only for the second and subsequent pages when results are paginated. The label page is API-implementation specific.
Resource Name	URI specific path	Statement specified as URI path segments. MAY be empty. MAY terminate with a slash (/) to designate a root for a subtree.
	URI specific page id	Page ID for a paginated results page. Resource Name value is returned in a Link: header response in a prior API request.
Resource Name Parameters	;r	;r specifies that the entire subtree of statements is to be returned. Applicable only when requesting a statement collection. Valid only when added to the end of the Resource Name.
	;size	;size=dd specifies a requested maximum number of statements for a statement collection returned in parts. Applicable only when requesting a statement collection. Valid only when added to the end of the Resource Name.
	;v	;v=vxx specifies a particular version of a statement at a node in the classification tree. Valid when added to any path segment in the Resource Name.
Request Headers	Accept:	Accept: header MAY be omitted. Only one Accept: header is permitted.
		application/vnd.ccss.standardstatement+JSON
		application/vnd.ccss.standardstatement+XML
		application/vnd.ccss.standardstatementcollection+JSON
		application/vnd.ccss.standardstatementcollection+XML
Request Body		Empty.
Response Headers	Content-Location:	Return the request message URI (REQUIRED).
	Content-Type:	application/vnd.ccss.standardstatement+JSON
		application/vnd.ccss.standardstatement+XML
		application/vnd.ccss.standardstatementcollection+JSON
		application/vnd.ccss.standardstatementcollection+XML
		application/JSON (for errors).
	Link:	To designate the next page in a paginated results set, use rel="next" in the header. Not included on the last page.
		To designate the next page in a paginated results set, use rel="prev" in the header. Not included with the first page.
	Set-Cookie:	SHALL NOT be used.

API Feature	Value	Notes
Response Status	200, OK	Statement(s) specified in Resource Name returned in the response body.
	404, Not Found	Statement(s) specified in Resource Name was not found in the statement data store. Response body is empty.
	Other	Error.
Response Body	JSON	JSON object for a standards statement.
		JSON object for a standards statement collection.
	XML	XML document for a standards statement.
		XML document for standards statement collection.
	JSON	Error message.
Authentication	NA	Not authenticated.
Encryption	Not Required	TLS not REQUIRED. TLS SHOULD NOT be used.
Errors		Return the most applicable error. Include error message in Response body.
Behavior	1. For a statement/ Resource Type: a. Validate the message URI and return the most applicable error if it is not valid b. Validate that the Resource Name and Accept: header correspond c. If the statement is not in the statement data store, return 404, Not Found d. Return the statement (or collection) in the requested media type i. Return 200, OK ii. Include a Content-Location: header iii. Include the Content-Type: header iv. For paginated results, include appropriate Link: rel="next" and Link: rel="prev" headers v. Include the statement (or collection) in the body 2. For a page/ Resource Type: a. Validate the message URI and return the most applicable error if it is not valid b. Validate that the Resource Name and Accept: header correspond c. If the pagination set has expired or been reset, return 409, Conflict d. Return the statement collection in the requested media type i. Return 200, OK ii. Include a Content-Location: header iii. Include the Content-Type: header iv. Include appropriate Link: rel="next" and Link: rel="prev" headers v. Include the statement collection in the body 3. Otherwise Error.	
Examples	GET /api/v1/statement/CCSS/Math/Content/./8/Math/G/A/1/a HTTP/1.1 Single statement.	
	GET /api/v1/statement/CCSS/Math/Content/./6/Math/NS/C/7/c/ HTTP/1.1 Collection of statements.	
	GET /api/v1/statement/CCSS/ELA;r HTTP/1.1 Entire subtree of statements.	
	GET /api/v1/statement/CCSS/Maths/Content/./13/./A/A/1 HTTP/1.1 Bad Resource Name.	
	GET /api/v1/page/995042e4-7f75-4e77-b83c-a79f24cb3a68 HTTP/1.1 Page request.	
Extensions		
Notes		

API Method: Standards Statement Retrieval Metainformation

API Feature	Value	Notes
Summary	Retrieve learning standards statement metainformation for: • An individual statement or version of a statement. • The collection of all statements at a given level in the statement classification tree. • The entire subtree of statements below a given level in the statement classification tree.	
HTTP Method	HEAD	
HTTP Protocol	HTTP/1.1	
API Base URI	api/	
Version	v1/	
Resource Type	statement/	statement/ is used to designate a statement or statement collection.
Resource Name	URI specific path	Statement specified as URI path segments. MAY be empty. MAY terminate with a slash (/) to designate a root for a subtree.
Resource Name Parameters	;r	;r specifies that the entire subtree of statements is to be returned. Applicable only when requesting a statement collection. Valid only when added to the end of the Resource Name.
	;size	;size=dd specifies a requested maximum number of statements for a statement collection returned in parts. Applicable only when requesting a statement collection. Valid only when added to the end of the Resource Name.
	;v	;v=vxx specifies a particular version of a statement at a node in the classification tree. Valid when added to any path segment in the Resource Name.
Request Headers	Accept:	Accept: header MAY be omitted. Only one Accept: header is permitted.
		application/vnd.ccss.standardstatement+JSON
		application/vnd.ccss.standardstatement+XML
		application/vnd.ccss.standardstatementcollection+JSON
		application/vnd.ccss.standardstatementcollection+XML
Request Body		Empty.
Response Headers	Content-Location:	Return the request message URI (REQUIRED).
	Content-Type:	application/vnd.ccss.standardstatement+JSON
		application/vnd.ccss.standardstatement+XML
		application/vnd.ccss.standardstatementcollection+JSON
		application/vnd.ccss.standardstatementcollection+XML
		application/JSON (for errors).
	Link:	To designate the next page in a paginated results set, use rel="next" in the header. Not included on the last page.
		To designate the next page in a paginated results set, use rel="prev" in the header. Not included with the first page.
	Set-Cookie:	SHALL NOT be used.
Response Status	200, OK	Statement(s) specified in Resource Name would be returned in the response body for a GET message.
	404, Not Found	Statement(s) specified in Resource Name was not found in the statement data store. Response body is empty.
	Other	Error.

API Feature	Value	Notes
Response Body	Empty	
Authentication	NA	Not authenticated.
Encryption	Not Required	TLS not REQUIRED. TLS SHOULD NOT be used.
Errors		Return the most applicable HTTP status code.
Behavior	1. For a statement/ Resource Type: a. Validate the message URI and return the most applicable error if it is not valid b. Validate that the Resource Name and Accept: header correspond c. If the statement is not in the statement data store, return 404, Not Found d. Return the statement (or collection) metainformation i. Return 200, OK ii. Include a Content-Location: header iii. Include the Content-Type: header iv. For paginated results, include appropriate Link: rel="next" and Link: rel="prev" headers v. Include an empty body 2. Otherwise error	
Examples	HEAD /api/v1/statement/CCSS/Math/Content/./8/Math/G/A/1/a HTTP/1.1 Metainformation for a single statement.	
	HEAD /api/v1/statement/CCSS/Math/Content/./6/Math/NS/C/7/c/ HTTP/1.1 Metainformation for a collection of statements.	
	HEAD /api/v1/statement/CCSS/ELA;r HTTP/1.1 Metainformation for an entire subtree of statements.	
	HEAD /api/v1/statement/CCSS/Maths/Content/./13/./A/A/1 HTTP/1.1 Metainformation for a bad Resource Name error.	
Extensions		
Notes	The HEAD message is not defined for a page in a paginated collection of results.	

API Method: Identifier Retrieval

API Feature	Value	Notes
Summary	Retrieve learning standards statements from the statement data store for: • An individual statement identifier. • The collection of all statements for all identifiers.	
HTTP Method	GET	
HTTP Protocol	HTTP/1.1	
API Base URI	api/	
Version	v1/	
Resource Type	id/	id/ is used to designate an identifier.
	page/	page/ is used to designate a page of a results. Used only for the second and subsequent pages when results are paginated. The label page is API-implementation specific.
Resource Name	identifier	Statement identifier. MAY be empty.
	URI specific page id	Page ID for a paginated results page. Resource Name value is returned in a Link: header response from a prior API request.
Resource Name Parameters	;http	;http specifies that only a web-resolvable locator for the requested identifier is returned. Applicable only when requesting a single identifier. Valid only when added to the end of the Resource Name.
	;loc	;loc specifies that only the locator for the requested identifier is returned. Applicable only when requesting a single identifier. Valid only when added to the end of the Resource Name.
	;size	;size=dd specifies a maximum page size for a statement collection returned in parts. Applicable only when requesting results for all identifiers. Valid only when added to the end of the Resource Name.
Request Headers	Accept:	Accept: header MAY be omitted. Only one Accept: header is permitted.
		application/vnd.ccss.standardstatement+JSON
		application/vnd.ccss.standardstatement+XML
		application/vnd.ccss.standardstatementcollection+JSON
		application/vnd.ccss.standardstatementcollection+XML
Request Body		Empty.
Response Headers	Content-Location:	For id/ return the request message URI. Otherwise return the message URI for the identified statement.
	Content-Type:	application/vnd.ccss.standardstatement+JSON
		application/vnd.ccss.standardstatement+XML
		application/vnd.ccss.standardstatementcollection+JSON
		application/vnd.ccss.standardstatementcollection+XML
		application/JSON (for errors).
	Link:	To designate the next page in a paginated results set, use rel="next" in the header. Not included on the last page.
		To designate the next page in a paginated results set, use rel="prev" in the header. Not included with the first page.
	Set-Cookie:	SHALL NOT be used.
Response Status	200, OK	Statement(s) for identifiers specified in Resource Name returned in the response body.

API Feature	Value	Notes
	404, Not Found	Identifier specified in Resource Name was not found in the statement data store. Response body is empty.
	Other	Error.
Response Body	JSON	JSON object for a standards statement.
		JSON object for a standards statement collection.
	XML	XML document for a standards statement.
		XML document for standards statement collection.
	JSON	Error message.
Authentication	NA	Not authenticated.
Encryption	Not Required	TLS not REQUIRED. TLS SHOULD NOT be used.
Errors		Return the most applicable error. Include error message in Response body.
Behavior	1. For an id/ Resource Type and an identifier Resource Name: a. Validate the message URI and return the most applicable error if it is not valid b. Validate that the Resource Name and Accept: header correspond c. If the identifier is not in the statement data store, return 404, Not Found d. Look up the identifier to find the statement e. Return the statement in the requested media type i. Return 200, OK ii. Include a Content-Location: header iii. Include the Content-Type: header iv. If ;loc or ;http is omitted include the statement in the body 2. For an id/ Resource Type and no Resource Name: a. Validate the message URI and return the most applicable error if it is not valid b. Validate that the Resource Name and Accept: header correspond c. Return the statement collection for all statements in the statement data store in the requested media type i. Return 200, OK ii. Include a Content-Location: header iii. Include the Content-Type: header i. For paginated results, include appropriate Link: rel="next" and Link: rel="prev" headers ii. Include the statement collection in the body 3. For a page/ Resource Type: a. Validate the message URI and return the most applicable error if it is not valid b. Validate that the Resource Name and Accept: header correspond c. If the pagination set has expired or been reset, return 409, Conflict d. Return the statement collection in the requested media type i. Return 200, OK ii. Include a Content-Location: header iii. Include the Content-Type: header iv. Include appropriate Link: rel="next" and Link: rel="prev" headers v. Include the statement collection in the body 4. Otherwise error	
Examples	GET /api/v1/id/610f436f266e4102a3b469a2b4d63da0 HTTP/1.1 Single statement for the ID.	
	GET /api/v1/id/5b37819c9b0d43eaafca4a86fb7091e7;loc HTTP/1.1 Locator for a single statement ID.	

API Feature	Value Notes
	GET /api/v1/id/5b37819c9b0d43eaafca4a86fb7091e7;http HTTP/1.1 Web-resolvable locator for a single statement ID.
	GET /api/v1/id/ HTTP/1.1 Collection of statements for all identifiers.
	GET /api/v1/id/5b37819c9b0d43eaafca4a86fb7091e7;r;v=2.5 HTTP/1.1 Bad Resource Name (includes a version with an id).
	GET /api/v1/page/995042e4-7f75-4e77-b83c-a79f24cb3a68 HTTP/1.1 Page request.
Extensions	
Notes	;loc is equivalent to a HEAD message.

API Method: Identifier Retrieval Metainformation

API Feature	Value	Notes
Summary	Retrieve learning standards statements metainformation for: • An individual statement identifier. • The collection of all statements for all identifiers.	
HTTP Method	HEAD	
HTTP Protocol	HTTP/1.1	
API Base URI	api/	
Version	v1/	
Resource Type	id/	id/ is used to designate an identifier.
Resource Name	identifier	Statement identifier. MAY be empty.
Resource Name Parameters	;http	;http specifies that only a web-resolvable locator for the requested identifier is returned. Applicable only when requesting a single identifier. Valid only when added to the end of the Resource Name.
	;loc	;loc specifies that only the locator for the requested identifier is returned. Applicable only when requesting a single identifier. Valid only when added to the end of the Resource Name.
	;size	;size=dd specifies a maximum page size for a statement collection returned in parts. Applicable only when requesting results for all identifiers. Valid only when added to the end of the Resource Name.
Request Headers	Accept:	Accept: header MAY be omitted. Only one Accept: header is permitted.
		application/vnd.ccss.standardstatement+JSON
		application/vnd.ccss.standardstatement+XML
		application/vnd.ccss.standardstatementcollection+JSON
		application/vnd.ccss.standardstatementcollection+XML
Request Body		Empty.
Response Headers	Content-Location:	For id/ return the request message URI. Otherwise return the message URI for the identified statement.
	Content-Type:	application/vnd.ccss.standardstatement+JSON
		application/vnd.ccss.standardstatement+XML
		application/vnd.ccss.standardstatementcollection+JSON
		application/vnd.ccss.standardstatementcollection+XML
		application/JSON (for errors).
	Link:	To designate the next page in a paginated results set, use rel="next" in the header. Not included on the last page.
		To designate the next page in a paginated results set, use rel="prev" in the header. Not included with the first page.
	Set-Cookie:	SHALL NOT be used.
Response Status	200, OK	Statement(s) for identifiers specified in Resource Name would be returned in the response body for a GET message.
	404, Not Found	Identifier specified in Resource Name was not found in the statement data store. Response body is empty.
	Other	Error.
Response Body	Empty	
Authentication	NA	Not authenticated.
Encryption	Not Required	TLS not REQUIRED. TLS SHOULD NOT be used.

API Feature	Value	Notes
Errors		Return the most applicable HTTP status code.
Behavior	1. For an id/ Resource Type and an identifier Resource Name: a. Validate the message URI and return the most applicable error if it is not valid b. Validate that the Resource Name and Accept: header correspond c. If the identifier is not in the statement data store, return 404, Not Found d. Look up the identifier to find the statement e. Return the statement metainformation i. Return 200, OK ii. Include a Content-Location: header iii. Include the Content-Type: header iv. Include an empty body 2. For an id/ Resource Type and no Resource Name: a. Validate the message URI and return the most applicable error if it is not valid b. Validate that the Resource Name and Accept: header correspond c. Return the statement collection metainformation i. Return 200, OK ii. Include a Content-Location: header iii. Include the Content-Type: header iv. For paginated results, include appropriate Link: rel="next" and Link: rel="prev" headers v. Include an empty body	
Examples	HEAD /api/v1/id/610f436f266e4102a3b469a2b4d63da0 HTTP/1.1 Single statement for the ID.	
	HEAD /api/v1/id/5b37819c9b0d43eaafca4a86fb7091e7;loc HTTP/1.1 Locator for a single statement ID.	
	HEAD /api/v1/id/ HTTP/1.1 Collection of statements for all identifiers.	
	HEAD /api/v1/id/5b37819c9b0d43eaafca4a86fb7091e7;r;v=2.5 HTTP/1.1 Bad Resource Name (includes a version with an id).	
	HEAD /api/v1/page/995042e4-7f75-4e77-b83c-a79f24cb3a68 HTTP/1.1 Page request.	
Extensions		
Notes	;loc is equivalent to a HEAD message.	
	The HEAD message is not defined for a page in a paginated collection of results.	

API Method: Standards Statement Publishing or Updating

API Feature	Value	Notes
Summary	Publish a new learning standards statement to the statement data store.	
	Update an existing learning standards statement to the statement data store.	
HTTP Method	PUT	
HTTP Protocol	HTTP/1.1	
API Base URI	api/	
Version	v1/	
Resource Type	statement/	statement/ is used to designate a statement.
Resource Name	URI specific path	Statement specified as URI path segments. SHALL NOT terminate with a slash (/).
Resource Name Parameters	;v	;v=vxx specifies a particular version of a statement at a node in the classification tree. Valid when added to any path segment in the Resource Name.
Request Headers	Authorization:	REQUIRED.
	Content-Type:	application/vnd.ccss.standardstatement+JSON
Request Body	JSON	JSON object for a standards statement.
Response Headers	Content-Location:	URI of the published statement (REQUIRED).
	Content-Type:	application/JSON (for errors).
Response Status	201, Entity Created	New statement was published to the statement data store. Response body is empty.
	204, No Content	Existing version of the statement was updated in the statement data store. Response body is empty.
	405, Method Not Allowed	Statement specified in Resource Name exists in the statement data store. Response body is empty.
	Other	Error.
Response Body	JSON	Error message.
Authentication	Required	Authentication is API-implementation specific.
Encryption	Required	TLS REQUIRED.
Errors		Return the most applicable error. Include error message in Response body.

API Feature	Value Notes
Behavior	1. For a statement/ Resource Type: a. Validate the message URI and return the most applicable error if it is not valid b. If the statement duplicates a statement is in the statement data store, return 405, Method Not Allowed c. Fully validate the statement to be published (well-formed JSON, schema validation, semantic validation) d. Augment the statement with any missing data, assign and home IDs, update relationships e. Publish/update the statement in the statement data store f. Return publication results g. If the statement is a statement or a new version of an existing statement i. If the statement is a new statement or a new version of an existing statement, Return 201, Entity Created ii. If the statement is an update of an existing statement, Return 204, No Content iii. Include a Content-Location: header iv. Include an empty body h. For a new statement or new version, publish the statement to the Learning Registry (optional) 2. Otherwise error
Examples	PUT /api/v1/statement/KSCCS/ELA-Literacy/CCRA/KS-LL/K-12/./././2 HTTP/1.1 Publish specified statement.
	PUT /api/v1/statement/CCSS/ELA-Literacy/Content/RL/7/./././5 HTTP/1.1 Update the specified statement.
	PUT /api/v1/statement/CCSS/Maths/Content/./13/./A/A/1 HTTP/1.1 Bad Resource Name.
Extensions	
Notes	The rules for govern the combinations of paths, versions, new publication and updating are presented in the *Resource Publication* Annex.
	Best practices dictate what is permitted as an update versus what requires a new statement version be published. Typically any substantial change requires a new version. The API implementation SHALL reject an update that should be a new version.

API Method: Standards Statement Deletion

API Feature	Value	Notes
Summary	Delete a learning standards statement from the statement data store.	
HTTP Method	DELETE	
HTTP Protocol	HTTP/1.1	
API Base URI	api/	
Version	v1/	
Resource Type	statement/	statement/ is used to designate a statement.
Resource Name	URI specific path	Statement specified as URI path segments. SHALL NOT terminate with a slash (/).
Resource Name Parameters	;v	;v=vxx specifies a particular version of a statement at a node in the classification tree. Valid when added to any path segment in the Resource Name.
Request Headers	Authorization:	REQUIRED.
Request Body		Empty.
Response Headers	Content-Type:	application/JSON (for errors).
Response Status	204, No Content	Statement specified in Resource Name was deleted from the statement data store. Response body is empty.
	404, Not Found	Statement specified in Resource Name was not found in the statement data store. Response body is empty.
	Other	Error.
Response Body	JSON	Error message.
Authentication	Required	Authentication is API-implementation specific.
Encryption	Required	TLS REQUIRED.
Errors		Return the most applicable error. Include error message in Response body.
Behavior	1. For a statement/ Resource Type: a. Validate the message URI and return the most applicable error if it is not valid b. If the statement is not in the statement data store, return 404, Not Found c. Delete the statement from the statement data store i. Return 204, No Content ii. Include an empty body 2. Otherwise error	
Examples	DELETE /api/v1/statement/CCSS/ELA-Literacy/CCRA/R/K-12/./././1 HTTP/1.1 Delete specified statement.	
	DELETE /api/v1/statement/CCSS/Maths/Content/./13/./A/A/1 HTTP/1.1 Bad Resource Name.	
Extensions		
Notes		

Implementation Considerations

An API implementation SHALL conform to [RFC 2616]. HTTP/1.1 places additional constraints on an API implementation and a client not described in this Specification.

A client MAY use any HTTP/1.1 feature not specifically excluded in this Specification. In particular:

- A client MAY include a CONTENT-MD5 request header.
- A client MAY include an ACCEPT-ENCODING request header.
- A response MAY include a CONTENT-MD5 response header.
- A response MAY include a CONTENT-ENCODING response header.

A client SHALL use the REQUIRED HTTP/1.1 features not specified in this Specification.

An API implementation SHALL use the REQUIRED HTTP/1.1 features not specified in this Specification.

Subsequent versions of the API SHOULD use simple integer version numbers, e.g., v2, v3.

Security Considerations

Producers including learning standards statement in JSON objects or XML documents or consumers accessing learning standards statement may want to consider the potential for unsolicited or malicious content and SHOULD take preventive measures to recognize such content and either identify it or not include it in their objects.

Producers of learning standards statements SHOULD take reasonable measures to make sure potentially malicious user input such as cross-site scripting attacks are not included in the learning standards statements or vocabularies they submit for publishing or updates.

API access methods MUST take reasonable measures to make sure potentially malicious ingested input is not distributed or emitted.

API users should be aware of the potential for malicious content where the attacker publishes objects with falsified property values with the intent of injecting malicious content, hiding or corrupting legitimate content, or misleading users.

JSON objects are subject to the same security considerations described in [RFC 4627]. The JSON objects may include URIs; see [RFC 3986] for security considerations. The JSON objects may include IRIs; see [RFC 3987] for security considerations.

The XML documents may include URIs; see [RFC 3986] for security considerations. The XML documents may include IRIs; see [RFC 3987] for security considerations.

HTTP methods are subject to the same security considerations described in [RFC 2616].

Authentication and authorization protocols and processes are subject to the same security considerations associated with the authentication and authorization protocols and processes used by the Publishing Agent.

Conformance

To conform to this Specification, an implementation of the API SHALL satisfy all normative requirements included in the Specification.

An implementation SHALL pass all conformance test cases (see Annex: *Conformance Test Cases*).

A minimally conforming implementation of the API SHALL fully implement the Standards Statement Retrieval API method. A minimally conforming implementation MAY implement all or parts of other API methods.

A complete conforming implementation of the API SHALL fully implement all API methods.

The Publishing Agent deployed in support of the Granular Identifiers and Metadata for CCSS (GIM CCSS) Project SHALL implement the API defined in this Specification. The Publishing Agent SHALL provide a minimally conforming implementation of the API. The Publishing Agent SHOULD provide a complete conforming implementation of the API.

Additional conformance requirements for a Publishing Agent or Publishing Organization are detailed in [Publishing Agent].

This Specification does not give additional conformance criteria for user agents that use the API.

Examples

Note: This section is informative.

Note: The examples are for illustrative purposes only. Statements and data used are examples.

Examples include:

- Retrieving a single standards statement.
- Retrieving a collection of standards statements.
- Retrieving a single standards statement given the statement's identifier, returned in XML.
- Retrieving all standards in the CCSS ELA Framework.
- Requesting a standards statement that does not exist.
- Retrieving a locator for a standards statement from the statement's identifier.
- Retrieving a web-resolvable locator for a standards statement from the statement's identifier.
- Publishing a new standards statement.
- Updating an existing standards statement.
- Deleting a standards statement.
- Paging through a collection of standards statements.
- An error example.

Retrieving a Standards Statement

The example illustrates how to retrieve a single standards statement.

The request illustrates:

- Specifying a single statement in the classification hierarchy (CCSS/Math/Content/./8/Math/G/A/1/a) in the API Resource URI.
- Using the default media type to return the JSON encoding.

Only a portion of the response is shown. It includes:

- Status [200, OK] indicating success.
- The Content-Type response header.
- The Location response header that is the locator of the statement.
- A portion of the JSON response.

API Request

```
GET /api/v1/statement/CCSS/Math/Content/./8/Math/G/A/1/a HTTP/1.1
Host: publishingagent.tld

```

API Response

```
HTTP/1.1 200 OK
Location: /api/v1/statement/CCSS/Math/Content/./8/Math/G/A/1/a
Content-Type: application/vnd.ccss.standardstatement+JSON
Content-Length: 0000

{
   "learningStandardsStatement": {
    "$schemaVersion": "GIM-CCSS 20130212",
    "$schemaLocation": "http: //publishingagent.tld/schema/learningstandardsstatement.json#",
      "identifiers": [
         {
            "identifier": {
               "idType": "GIM Path",
               "id": "CCSS/Math/Content/./8/Math/G/A/1/a"
            }
         }
…
}
```

Note: API request and response validated: Not validated.

Retrieving a Collection of Standards Statement

The example illustrates how to retrieve a collection of standards statements.

The request illustrates:

- Specifying a single statement in the classification hierarchy (CCSS/Math/Content/./6/Math/NS/C/7/c/) in the API Resource URI.
- Adding a slash (/) to the request to indicate that the statements below the specified root are to be returned.
- Using an Accept header in the request.

Only a portion of the response is shown. It includes:

- Status [200, OK] indicating success.
- The Content-Type response header.
- The Location response header that is the locator of the collection.
- A portion of the JSON response.
- Note, there are only 2 granular statements in the collection – the response is for the statements "below" the specified root, and does not include the root statement specified in the message URI.

API Request

```
GET /api/v1/statement/CCSS/Math/Content/./6/Math/NS/C/7/c/ HTTP/1.1
Host: publishingagent.tld
Accept: application/vnd.ccss.standardstatementcollection+JSON

```

API Response

```
HTTP/1.1 200 OK
Location: api/v1/statement/CCSS/Math/Content/./6/Math/NS/C/7/c/
Content-Type: application/vnd.ccss.standardstatementcollection+JSON
Content-Length: 0000

{
  "learningStandardsStatementCollection": {
   "$schemaVersion": "GIM-CCSS 20130212",
   "$schemaLocation": "http: //publishingagent.tld/schema/learningstandardsstatementcollection.json",
    "totalStatements": 2,
    "statements": [
      {
        "learningStandardsStatement": {
          "identifiers": [
            {
              "identifier": {
                "idType": "GIM Path",
                "id": "CCSS/Math/Content/./6/Math/NS/C/7/c/1"
              }
            }
...
}
```

Note: API request and response validated: Not validated.

Retrieving a Standards Statement in XML from the Statement Identifier

The example illustrates how to retrieve a single standards statement given the statement's identifier.

The request illustrates:

- Specifying the statement identifier (610f436f266e4102a3b469a2b4d63da0) in the API Resource URI.
- Using an Accept header in the request.

Only a portion of the response is shown. It includes:

- Status [200, OK] indicating success.
- The Content-Type response header.
- The Location response header that is the locator of the statement (not the identifier).
- A portion of the XML response.

API Request

```
GET /api/v1/id/610f436f266e4102a3b469a2b4d63da0 HTTP/1.1
Host: publishingagent.tld
Accept: application/vnd.ccss.standardstatement+XML

```

API Response

```
HTTP/1.1 200 OK
Location: api/v1/statement/CCSS/Math/Content/./6/Math/NS/C/7/c/
Content-Type: application/vnd.ccss.standardstatement+XML
Content-Length: 0000

<xml>
…
</xml>
```

Note: API request and response validated: Not validated.

Retrieving all Standards Statement in the ELA Framework

The example illustrates how to retrieve the entire collection of statements in the CCSS ELA Framework.

The request illustrates:

- Specifying the framework (CCSS/ELA) in the API Resource URI.
- Including a ;r (recursive subtree) parameter in the final path segment to request all statements be returned.
- Using an Accept header in the request.

Only a portion of the response is shown. It includes:

- Status [200, OK] indicating success.
- The Content-Type response header.
- The Location response header that is the locator of the collection.
- A portion of the JSON response.
- Note, the results would include the root of the subtree (CCSS/ELA) along with all the statements below the root (CCSS/ELA/;r would omit the root from the results).

API Request

```
GET /api/v1/statement/CCSS/ELA;r HTTP/1.1
Host: publishingagent.tld
Accept: application/vnd.ccss.standardstatementcollection+JSON

```

API Response

```
HTTP/1.1 200 OK
Location: api/v1/statement/CCSS/ELA;r
Content-Type: application/vnd.ccss.standardstatementcollection+JSON
Content-Length: 0000

{
   "learningStandardsStatementCollection": {
     "$schemaVersion": "GIM-CCSS 20130212",
     "$schemaLocation": "http: //publishingagent.tld/schema/learningstandardsstatementcollection.json",
       "totalStatements":  929,
       "statements": [
          {
             "learningStandardsStatement": {
                "identifiers": [
                   {
                      "identifier": {
                         "idType": "GIM Path",
                         "id": "CCSS/ELA"
                      }
                   }
…
}
```

Note: API request and response validated: Not validated.

Requesting a Standards Statement That Does Not Exist

The example illustrates requesting a standards statement that does not exist.

The request illustrates:

- An invalid statement (CCSS/Maths/Content/./13/Math/A/A/1) in the API Resource URI.
- Using the default media type to return the JSON encoding.

The full response is shown (a detailed error response is not returned). It includes:

- Status [404, Not Found] indicating the statement does not exist.

API Request

```
GET /api/v1/statement/CCSS/Maths/Content/./13/./A/A/1 HTTP/1.1
Host: publishingagent.tld

```

API Response

```
HTTP/1.1 404 Not Found

```

Note: API request and response validated: Not validated.

Retrieving a Standards Statement Locator from the Statement Identifier

The example illustrates how to retrieve a single standards locator given the statement's identifier.

The request illustrates:
- Specifying the statement identifier (03073da6c76447ada8439e49862ab660) in the API Resource URI.
- Including a ;loc (locator) parameter in the final path segment to request the statement's locator be returned.

The full response is shown. It includes:
- Status [200, OK] indicating success.
- The Location response header that is the locator of the statement (not the identifier).

API Request

```
GET /api/v1/id/03073da6c76447ada8439e49862ab660;loc HTTP/1.1
Host: publishingagent.tld

```

API Response

```
HTTP/1.1 200 OK
Location: api/v1/statement/CCSS/Math/Content/./HS/S/CP/A/4/./2

```

Note: API request and response validated: Not validated.

Retrieving a Standards Statement Web-Resolvable Locator from the Statement Identifier

The example illustrates how to retrieve a single standards web-resolvable (http) locator given the statement's identifier.

The request illustrates:
- Specifying the statement identifier (03073da6c76447ada8439e49862ab660) in the API Resource URI.
- Including an ;http (web-resolvable locator) parameter in the final path segment to request the statement's web-resolvable locator be returned.

The full response is shown. It includes:
- Status [200, OK] indicating success.
- The Location response header that is the locator of the statement (not the identifier).

API Request

```
GET /api/v1/id/03073da6c76447ada8439e49862ab660;http HTTP/1.1
Host: publishingagent.tld

```

API Response

```
HTTP/1.1 200 OK
Location: http://publishingagent.tld/CCSS/Math/Content/./HS/S/CP/A/4/./2

```

Note: API request and response validated: Not validated.

Publishing a New Standards Statement

The example illustrates how to publish a new standards statement to the statement data store.

The request illustrates:

- Specifying a single statement in the classification hierarchy (KSCCS/ELA-Literacy/CCRA/KS-LL/K-12/././././2) in the API Resource URI. The example shows a state 15% extension.
- Using an Authorization header in the request to pass the authentication credentials.
- Specifying the type of the attached JSON content.
- The new statement in the body of the request (only a portion of the statement is shown).

The full response is shown. It includes:

- Status [201, Entity Created] indicating success.
- The Location response header that is the locator of the published statement.

API Request

```
PUT /api/v1/statement/KSCCS/ELA-Literacy/CCRA/KS-LL/K-12/././././2 HTTP/1.1
Host: publishingagent.tld
Authorization:  API-specific-authentication-credentials
Content-Type: application/vnd.ccss.standardstatement+JSON
Content-Length: 0000

{
  "learningStandardsStatement": {
   "$schemaVersion": "GIM-CCSS 20130212",
   "$schemaLocation": "http: //publishingagent.tld/schema/learningstandardsstatement.json#",
    "identifiers": [
      {
        "identifier": {
          "idType": "GIM Path",
          "id": "KSCCS/ELA-Literacy/CCRA/KS-LL/K-12/././././2"
        }
      }
    ]
  },
  "classifiers": {
    "taxons": {
      "initiative": "KSCCS",
      "framework": "ELA-Literacy",
      "set": "CCRA",
      "strand": "KS-LL",
      "grade": "K-12",
      "standard": "2"
    },
    "statementType": "Standard"
  }
…
}
```

API Response

```
HTTP/1.1 201 Entity Created
Location: api/v1/statement/KSCCS/ELA-Literacy/CCRA/KS-LL/K-12/./././2

```

Note: API request and response validated: Not validated.

Updating an Existing Standards Statement

The example illustrates how to update and version an existing standards statement to the statement data store.

The request illustrates:

- Specifying a single statement in the classification hierarchy (CCSS/ELA-Literacy/Content/RL/7/./././5) in the API Resource URI. Note, the request does not include the version parameter in the request message URI. This implies a new version of the statement is to be created.
- Using an Authorization header in the request to pass the authentication credentials.
- Specifying the type of the attached JSON content
- The updated statement in the body of the request (only a portion of the statement is shown).

The full response is shown. It includes:

- Status [201, Entity Created] indicating success.
- The Location response header that is the locator of the published statement. The result includes the version parameter.

API Request

```
PUT /api/v1/statement/CCSS/ELA-Literacy/Content/RL/7/./././5 HTTP/1.1
Host: publishingagent.tld
Authorization:  API-specific-authentication-credentials
Content-Type: application/vnd.ccss.standardstatement+JSON
Content-Length: 0000

{
   "learningStandardsStatement": {
      "$schemaVersion": "GIM-CCSS 20130212",
      "$schemaLocation": "http: //publishingagent.tld/schema/learningstandardsstatement.json#",
      "identifiers": [
         {
            "identifier": {
               "idType": "GIM UUID",
               "id": "4691b0642c3a4d42a4264b11ee0f8be5"
            }
         },
         {
            "identifier": {
               "idType": "GIM Path",
               "id": "CCSS/ELA-Literacy/Content/RL/7/./././5;v=2.0"
            }
         }
…
}
```

API Response

```
HTTP/1.1 201 Entity Created
Location: api/v1/statement/CCSS/ELA-Literacy/Content/RL/7/./././5;v=2.0

```

Note: API request and response validated: Not validated.

Deleting a Standards Statement

The example illustrates how to delete a standards statement from the statement data store. The example illustrates:

- Specifying a single statement in the classification hierarchy (CCSS/ELA-Literacy/CCRA/R/K-12/././1) in the API Resource URI.
- Using an Authorization header in the request to pass the authentication credentials.

The full response is shown. It includes:

- Status [204, No Content] indicating success.

API Request

```
DELETE /api/v1/statement/CCSS/ELA-Literacy/CCRA/R/K-12/././1 HTTP/1.1
Host: publishingagent.tld
Authorization:  API-specific-authentication-credentials

```

API Response

```
HTTP/1.1 204 No Content

```

Note: API request and response validated: Not validated.

Paging Through a Collection of Standards Statements

The example illustrates how to page through a collection of standards statements. The example illustrates:

- Specifying a single statement in the classification hierarchy (CCSS/Math/Content) in the message URI.
- Adding a slash (/) to the request to indicate that the statements below the specified root are to be returned.
- Including a ;size (size) parameter in the final path segment to request results be paginated.
- Using an Accept header in the request.

Only a portion of the response is shown. For the first page in the results set, it includes:

- Status [200, OK] indicating success.
- The Content-Type response header.
- The Location response header that is the locator of the collection.
- A Link response header for the next page of results (the response is the first page).
- A portion of the JSON response. Note, in the example, the returned collection is smaller than the requested size.

API Request

```
GET /api/v1/statement/CCSS/Math/Content;size=100 HTTP/1.1
Host: publishingagent.tld
Accept: application/vnd.ccss.standardstatementcollection+JSON

```

API Response

```
HTTP/1.1 200 OK
Location: api/v1/statement/CCSS/Math/Content/
Content-Type: application/vnd.ccss.standardstatementcollection+JSON
Link: <api/v1/page/995042e4-7f75-4e77-b83c-a79f24cb3a68> rel="next"
Content-Length: 0000

{
  "learningStandardsStatementCollection": {
    "$schemaVersion": "GIM-CCSS 20130212",
    "$schemaLocation": "http: //publishingagent.tld/schema/learningstandardsstatementcollection.json",
      "totalStatements": 64,
      "statements": [
        {
          "learningStandardsStatement": {
            "identifiers": [
              {
                "identifier": {
                  "idType": "GIM Path",
                  "id": "CCSS/Math/Content/./"
                }
              }
...
}
```

The user can then request the next page of results. For the purposes of the example, assume that the API implementation resets pagination before the user requests the second page of results.

The example illustrates:

- Using the page link (/api/v1/page/995042e4-7f75-4e77-b83c-a79f24cb3a68) in the message URI.

The full response is shown. It includes:

- Status [409, Conflict] indicating reset.
- The Location response header that can be used to restart the entire request and recover from the error.
- The Content-Type response header for the error.
- The error message.

API Request

```
GET /api/v1/page/995042e4-7f75-4e77-b83c-a79f24cb3a68 HTTP/1.1
Host: publishingagent.tld
Accept: application/vnd.ccss.standardstatementcollection+JSON

```

API Response

```
HTTP/1.1 409 Conflict
Location: api/v1/statement/CCSS/Math/Content/
Content-Type: application/JSON
Content-Length: 0000

{
  "error": {
    "httpStatusCode": "409",
    "httpStatus": "409 Conflict ",
    "apiErrorCode": "Pagination-0001",
    "apiErrorDescription": "Pagination reset",
    "apiRequest": "GET /api/page/995042e4-7f75-4e77-b83c-a79f24cb3a68 HTTP/1.1"
  }
}
```

Note: API requests and responses validated: Not validated.

Error Example

The example illustrates what happens when there in an error in the API message. The example is based on statement publication, with the assumption that the statement contains an error detected during input validation.

The request illustrates:

- Specifying a single statement in the classification hierarchy (KSCCS/ELA-Literacy/CCRA/KS-LL/K-12/././././2) in the API Resource URI.
- Using an Authorization header in the request to pass the authentication credentials.
- Specifying the type of the attached JSON content.
- The new statement in the body of the request (only a portion of the statement is shown).

The full response is shown. It includes:

- Status [400, Bad Request] indicating an error.
- An error message detailing the validation error.

API Request

```
PUT /api/v1/statement/KSCCS/ELA-Literacy/CCRA/KS-LL/K-12/./././2 HTTP/1.1
Host: publishingagent.tld
Authorization:  API-specific-authentication-credentials
Content-Type: application/vnd.ccss.standardstatement+JSON
Content-Length: 0000

{
  "learningStandardsStatement": {
   "$schemaVersion": "GIM-CCSS 20130212",
   "$schemaLocation": "http: //publishingagent.tld/schema/learningstandardsstatement.json#",
    "identifiers": [
      {
        "identifier": {
          "idType": "GIM Path",
          "id": "KSCCS/ELA-Literacy/CCRA/KS-LL/K-12/./././2"
        }
…
}
```

API Response

```
HTTP/1.1 400 Bad Request
Content-Type: application/JSON
Content-Length: 0000

{
  "error": {
    "httpStatusCode": "400",
    "httpStatus": "400 Bad Request ",
    "apiErrorCode": "Validation-1312",
    "apiErrorDescription": " Taxon value is not valid for the specified taxon vocabulary ",
    "apiRequest": "PUT /api/v1/statement/KSCCS/ELA-Literacy/CCRA/KS-LL/K-12/./././2 HTTP/1.1"
  }
}
```

Note: API request and response validated: Not validated.

API Design Decisions

Note: This section is informative.

The API includes the HTTP HEAD message for completeness. The method could be deleted without loss of core functionality.

The API does not support discovery of API versions or resource types; request URIs cannot terminate after the version or resource types part of the path. For simplicity in API design, results format to return the available API versions or resource types has not been defined. By designing an appropriate results format, the API could be extended to return the available API versions or available resource types in the statement data store.

While not a fully RESTful design, the API includes the API version in the request URI. Alternatively, the API version could be included in a custom HTTP header. Including versions in the request is common practice; custom headers may not be passed through proxies and firewalls.

Both the API URI Base and API version are included in the request URI. Alternatively, they could have been included in the host name.

The API does not include API service discovery. Supporting service discovery is not an essential feature. The API can be extended to support service discovery without changing other functionality. The API could support service discovery through a GET message with an Accept header of media type for a service description, e.g., application/xrds. The request would return a service description document for the API.

Validation of a standards statement does not check that all relationship links are valid. There is no requirement that all links be valid, or that statements are published in an order that permits link validation; a statement may link to another statement that is not yet published. Enforcing link checking as part of validation would complicate the publishing and updating methods and overall workflow. An additional API method could be created to check the entire statement data store or an individual statement to validate all links.

The API does not include a mechanism to retrieve a part of a standards statement; only an entire statement is returned. Retrieving a part of a statement would add complexity to the API. The URI path could be extended to specify an object or element in the standards statement to retrieve a part of the statement.

The API does not provide a query mechanism. Query would add complexity to the API.

The API does not provide a mechanism to sort or order the results in a learning standards statement collection. Sorting would add complexity to the API.

The API design does not specify the authentication or authorization models and protocols. The details of authentication and authorization are left to the Publishing Agent. This allows different Publishing Agents to use different authentication or authorization models and protocols.

The API design does not specify the form of URIs returned in Link headers for paginated results. The API design may design URIs to encode state information to support cache control and maintaining pagination sets.

The API does not include an indexed form to access a particular part of a collection. Pagination is the only mechanism provided to access a part of a collection.

Extending the PUT (or POST) messages to include the JSON document as a URI parameter of the HTTP URI instead of in the body of the message has not been included. JSON URI serialization such as [Rison] could be used to pass the learning standards statement as the value of a parameter labeled json.

Rules for validating requests prioritize the Resource URI over the Accept header, e.g., if the Resource URI does not end in a slash (does not indicate a collection), an Accept header requesting a collection media type is an error. The API could be more permissive, allowing cases such as the above, not requiring strict alignment of the URI and header.

The API does not include a media type to return a standards statement in an HTML formatted page view.

The API includes options to specify versions, page sizes and recursive retrieval as part of the Resource URI. As the information model supports versioning at any level in the statement hierarchy, query parameters cannot be used to specify the versions; versions can be specified for **any** part of the statement path, e.g.,

```
GET /api/v1/statement/CCSS;v=2/Math/Content/6;v=3/NS/C;v=2;r;size=100
```

If the information model only permitted versions only of the terminal statement, and not each part of the statement hierarchy, the URI query notation could be used, e.g.,

```
GET /api/v1/statement/CCSS/Math/Content/6/NS/C?v=2&r=t&size=100
```

but this was rejected as it does not meet the requirements.

The ";" was chosen as one of the acceptable URI sub-delims [RFC 3986] to delimit the version parameter from the path segment. Other sub-delims e.g., "!", "$", "&", "*", "+", or the unreserved character "~" could have been chosen to delimit the parameter from the path segment. The entire set of potential sub-delims and unreserved characters is not available as some are used in the names of the parts of the path, e.g., "-" and "_".

For consistency, the ";" syntax and path parameters are used for all statement parameters versus using path parameters only for versions and using query parameters for size and recursion, e.g.,

```
GET /api/v1/statement/CCSS;v=2/Math/Content/6;v=3/NS/C;v=2?r=t&size=100
```

was rejected as a design choice.

The alternative syntax of using only query parameters, e.g.,

```
GET/api/v1/statement/standard?initiative=CCSS&standard=3&strand=w&framework=ELA&strand=w&
grade=6&inititiveversion=2015&standardversion=2
```

was rejected as a design choice.

Alternative design approaches, including WebDAV and AtomPub were rejected.

Glossary

Data Store: A *data store* is a tool for the storage and management of data. The data source exposes services allowing access to that data by other parties. [PILIN]

Digital Representation: The digital form of a *standards statement*, including both provenance metadata and the description of all other characteristics of the statement.

Granular Identifier: This concept does not exist within the technical approach. All identifiers are equal; none is more or less granular than any other.

Granular Statement: A *standards statement* derived from the decomposition or mapping of an existing standards statement to identify existing concepts within the statement for the purpose of aligning learning resources and assessment items to part of a statement.

Identifier: An identifier is the association of a *name* with a *thing*. A name may only be associated with one thing at any time, and the name is said to *identify* the thing. [PILIN]

Identifier Management System: An *identifier management system* is a collection of definitions, *information models*, *policies*, and *data stores*, used to manage *identifiers*. An identifier management system has a defined operating scope, and an *authority* acting as its owner. [PILIN]

Information Model: An *information model* is a model of *things* in a domain, their properties, and the relations between them. The choice of what things to *identify* in an *identifier management system* is informed by an information model. [PILIN]

Internet-Resolvable: See *Resolvable.*

Label: A *label* is a symbol that can potentially be used as a *name*. In an *identifier management system*, labels are typically strings. [PILIN]

Locator: A locator is a string giving the location of a digital object in a *data store*, and can be used as a retrieval key to gain access to the object. A URL is an example of a locator, although not all http: URIs are locators. A locator is specific to a *data store,* and cannot be used to access an instance of the digital object in a different *data store*. A locator can be used as an *identifier*; but it will usually not be *persistent*. Persistent identifiers often *resolve* to the current locator(s) of the *thing identified*. This uncouples the persistent identification of a resource from the current retrieval key for the resource. [PILIN]

Name: A *name* is the association of a *label* with a *context*. [PILIN]

Publish: *Publishing* is the process of making the digital representation of a *standards statement* available on the Internet for others to use.

Publishing Agent: The *Publishing Agent* is the deployed computing environment used to *publish* standards statements.

Publishing Organization: The *Publishing Organization* is the organization or entity that operates the *Publishing Agent*.

Resolve: An identifier is *resolved* by providing information on how to access the *thing* it identifies. This information is the *resolution* of the identifier. An identifier is *Internet-Resolvable* if the information on how to access the thing identified can be requested and consumed through a well-defined Internet application protocol, and *Web-Resolvable* if that protocol is a defined web application layer protocol. Resolution in general operates on *association data,* stored, managed and maintained with the identifier in an *identifier management system*. [PILIN]

Standards Statement: The narrative statement for a single educational standard that defines "what students should understand and be able to do." Within CCSS a standards statement or the components of a standards statement are the most granular statements.

Statement: See *Standards Statement*.

Statement Originator: The *Statement Originator* is the organization or entity responsible for the development of a *standards statement*. Statement originators include CCSSO, SETDA and states.

Web-Resolvable: See *Resolvable.*

Note: Some terms come from the PILIN Glossary, as noted. Some PILIN definitions have been simplified. Words in italics in a gloss are defined elsewhere in the glossary.

Normative References

For dated references, only the edition cited applies. For undated references, the most recent edition applies.

[GIM CCSS JSON] *Learning Standards Digital Representation Specification: JSON Serialization Schemata,* Granular Identifiers and Metadata for CCSS (GIM CCSS) Project.

[GIM CCSS XML] *Learning Standards Digital Representation Specification: XML Serialization Schemata,* Granular Identifiers and Metadata for CCSS (GIM CCSS) Project.

[JSON Schema Core] Galiegue, F., Zyp, K., and Court, G., "JSON Schema: core definitions and terminology json-schema-core", January 2013.
[http://json-schema.org/latest/json-schema-core.html]

[JSON Schema Validation] Galiegue, F., Zyp, K., and Court, G., "JSON Schema: interactive and non interactive validation json-schema-validation", January 2013.
[http://json-schema.org/latest/json-schema-validation.html]

[Publishing Agent] *Learning Standards Digital Representation Specification: Publishing Agent Design and Requirements*, Granular Identifiers and Metadata for CCSS (GIM CCSS) Project

[RFC 2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", IETF RFC 2119, March 1997.
[http://tools.ietf.org/html/rfc2119]

[RFC 2616] Fielding, R., "Hypertext Transfer Protocol – HTTP/1.1", IETF RFC 2616, June 1999.
[http://tools.ietf.org/html/rfc2616]

[RFC 3339] Klyne, G., "Date and Time on the Internet: Timestamps", IETF RFC 3339, July 2002.
[http://tools.ietf.org/html/rfc3339]

[RFC 3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI)", IETF RFC 3986, January 2005.
[http://tools.ietf.org/html/rfc3986]

[RFC 3987] Duerst, M., and Suignard, M., "Internationalized Resource Identifiers (IRIs)", IETF RFC 3987, January 2005.
[http://tools.ietf.org/html/rfc3987]

[RFC 4288] Freed, N., and Klensin, J., "Media Type Specifications and Registration Procedures", IETF RFC 4288, December 2005.
[http://tools.ietf.org/html/rfc4288]

[RFC 4289] Freed, N., and Klensin, J., "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", IETF RFC 4288, December 2005.
[http://tools.ietf.org/html/rfc4289]

[RFC 4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", IETF RFC 4627, July 2006.
[http://tools.ietf.org/html/rfc4627]

[RFC 5246] Dierks, T., and Rescorla, E., "The Transport Layer Security (TLS) Protocol, Version 1.2", IETF RFC 5246, August 2008.
[http://tools.ietf.org/html/rfc5246]

[RFC 5988] Nottingham, M., "Web Linking", IETF RFC 5988, October 2010.
[http://tools.ietf.org/html/rfc5988]

[XML] Bray, T., et al., "Extensible Markup Language (XML) 1.1", Second Edition, *W3C Recommendation*, August 2006.
[http://www.w3.org/TR/xml11/]

[XSD 1] Thompson, H., et al., "XML Schema Definition Language (XSD) 1.1 Part 1: Structures", *W3C Recommendation*, April 2012.
[http://www.w3.org/TR/xmlschema11-1/]

[XSD 2] Peterson, D., et al., "XML Schema Definition Language (XSD) 1.1 Part 2: Data Types", *W3C Recommendation*, April 2012.
[http://www.w3.org/TR/xmlschema11-2/]

Informative References

[JSON Schema Generator] *JSON Schema.net.*
[http://www.jsonschema.net/]

[JSON Schema Validator] JSON Schema Validator.
[http://json-schema-validator.herokuapp.com/syntax.html]

[JSON Validator] *JSONLint – The JSON Validator.*
[http://jsonlint.com/]

[REST] Representational State Transfer (REST).
[http://en.wikipedia.org/wiki/REST]

[Rison] Rison - Compact Data in URIs.
[http://mjtemplate.org/examples/rison.html]

[PILIN] *PILIN Ontology for Identifiers and Identifier Services.*
[http://resolver.net.au/hdl/102.100.272/ G9JR4TLQH]

[Scope] *Scope, Technical Requirements, Approaches, and Recommendations*, Granular Identifiers and Metadata for CCSS (GIM CCSS) Project.

Annex: Resource URIs, Resource Parameters and Accept Headers

A GET message MAY be accompanied with an Accept header that specifies the requested media type. The following rules SHALL be used to determine the result.

The rules SHALL take precedence over any other stated behavior in this Specification.

API Resource base URI is omitted

- Result SHALL be a Request-0002 error (URI omitted) [400, Bad Request]

API Resource base URI is not API

- Result SHALL be a Request-0003 error (URI not valid) [400, Bad Request]

API Version is omitted

- Result SHALL be a Request-0004 error (Version omitted) [400, Bad Request]

API Version is not v1

- Result SHALL be a Request-0005 error (Version not supported) [501, Not Implemented]

Resource Type is omitted

- Result SHALL be a Request-0006 error (Type omitted) [400, Bad Request]

Resource Type is not one of statement/, id/ or vocabulary/

- Result SHALL be a Request-0007 error (Type not valid) [400, Bad Request]

Resource Type is statement/

- Resource Name is omitted
  AND
- Resource URI does **NOT** include a ;r (recursive subtree) parameter
- An included ;loc (location) parameter is an error
    - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- An included ;http (web-resolvable locator) parameter is an error
    - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- Included ;v (version) parameters is an error
    - Result SHALL be a Request-0109 error (Versioning not applicable) [406, Not Acceptable]

  - Accept header is omitted
    - Result SHALL be the collection of all statements at the root level in the statement data store as application/vnd.ccss.standardstatementcollection+JSON
  - Accept header is application/vnd.ccss.standardstatementcollection+JSON
    - Result SHALL be the collection of all statements at the root level in the statement data store as application/vnd.ccss.standardstatementcollection+JSON
  - Accept header is application/vnd.ccss.standardstatement+JSON
    - Result SHALL a Request-0102 error (requesting an item for a collection path)
  - Accept header is application/vnd.ccss.standardstatementcollection+XML
    - Result SHALL be the collection of all statements at the root level in the statement data store as application/vnd.ccss.standardstatementcollection+XML
  - Accept header is application/vnd.ccss.standardstatement+JSON
    - Result SHALL be a Request-0102 error (Requesting an item for a collection path) [406, Not Acceptable]

- Accept header is any other media type
  - Result SHALL be a Request-0103 error (Requesting a non statement media type) [406, Not Acceptable]

Resource Type is statement/

- Resource Name is omitted
  AND
- Resource URI includes a ;r (recursive subtree) parameter at the end of the URI
- An included ;loc (location) parameter is an error
  - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- An included ;http (web-resolvable locator) parameter is an error
  - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- Included ;v (version) parameters is an error
  - Result SHALL be a Request-0109 error (Versioning not applicable) [406, Not Acceptable]

  - Accept header is omitted
    - Result SHALL be the collection of all statements in the statement data store as application/vnd.ccss.standardstatementcollection+JSON
  - Accept header is application/vnd.ccss.standardstatementcollection+JSON
    - Result SHALL be the collection of all statements in the statement data store as application/vnd.ccss.standardstatementcollection+JSON
  - Accept header is application/vnd.ccss.standardstatement+JSON
    - Result SHALL a Request-0102 error (requesting an item for a collection path)
  - Accept header is application/vnd.ccss.standardstatementcollection+XML
    - Result SHALL be the collection of all statements in the statement data store as application/vnd.ccss.standardstatementcollection+XML
  - Accept header is application/vnd.ccss.standardstatement+JSON
    - Result SHALL be a Request-0102 error (Requesting an item for a collection path) [406, Not Acceptable]
  - Accept header is any other media type
    - Result SHALL be a Request-0103 error (Requesting a non statement media type) [406, Not Acceptable]

Resource Type is statement/

- Resource Name is a statement classification path without a terminal slash
  AND
- Resource URI does **NOT** include a ;r (recursive subtree) parameter in the final path segment
- An included resource URI ;size (size) parameter is an error
  - Result SHALL be a Request-0106 error (Requesting paginated results for a single item) [406, Not Acceptable]
- An included ;loc (location) parameter is an error
  - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- An included ;http (web-resolvable locator) parameter is an error
  - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) \ [406, Not Acceptable]

  - Accept header is omitted

    - Result SHALL be the requested statement as application/vnd.ccss.standardstatement+JSON
  - Accept header is application/vnd.ccss.standardstatementcollection+JSON
    - Result SHALL be a Request-0101 error (Requesting a collection for an item path) [406, Not Acceptable]
  - Accept header is application/vnd.ccss.standardstatement+JSON
    - Result SHALL be the requested statement as application/vnd.ccss.standardstatement+JSON
  - Accept header is application/vnd.ccss.standardstatementcollection+XML
    - Result SHALL be a Request-0101 error (Requesting a collection for an item path) [406, Not Acceptable]
  - Accept header is application/vnd.ccss.standardstatement+XML
    - Result SHALL be the requested statement as application/vnd.ccss.standardstatement+XML
  - Accept header is any other media type
    - Result SHALL be a Request-0103 error (Requesting a non statement media type) [406, Not Acceptable]

Resource Type is statement/

- Resource Name is a statement classification path without a terminal slash
  AND
- Resource URI includes a ;r (recursive subtree) parameter in the final path segment
- An included ;loc (location) parameter is an error
    - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- An included ;http (web-resolvable locator) parameter is an error
    - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]

  - Accept header is omitted
    - Result SHALL be the collection of all statements in the subtree rooted at the path (including the root) as application/vnd.ccss.standardstatementcollection+JSON
  - Accept header is application/vnd.ccss.standardstatementcollection+JSON
    - Result SHALL be the collection of all statements in the subtree rooted at the path (omitting the root) as application/vnd.ccss.standardstatementcollection+JSON
  - Accept header is application/vnd.ccss.standardstatement+JSON
    - Result SHALL be a Request-0102 error (Requesting an item for a collection path) [406, Not Acceptable]
  - Accept header is application/vnd.ccss.standardstatementcollection+XML
    - Result SHALL be the collection of all statements in the subtree rooted at the path (including the root) as application/vnd.ccss.standardstatementcollection+XML
  - Accept header is application/vnd.ccss.standardstatement+XML
    - Result SHALL be a Request-0102 error (Requesting an item for a collection path) [406, Not Acceptable]
  - Accept header is any other media type
    - Result SHALL be a Request-0103 error (Requesting a non statement media type) [406, Not Acceptable]

Resource Type is statement/

- Resource Name is a statement classification path with a terminal slash
  AND
- Resource URI does **NOT** include a ;r (recursive subtree) parameter in the final path segment
- An included ;loc (location) parameter is an error

        - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- An included ;http (web-resolvable locator) parameter is an error
        - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]

    - Accept header is omitted
        - Result SHALL be the collection of all statements in the level immediately below the root specified in the path as application/vnd.ccss.standardstatementcollection+JSON
    - Accept header is application/vnd.ccss.standardstatementcollection+JSON
        - Result SHALL be the collection of all statements in the level immediately below the root specified in the path as application/vnd.ccss.standardstatementcollection+JSON
    - Accept header is application/vnd.ccss.standardstatement+JSON
        - Result SHALL be a Request-0102 error (Requesting an item for a collection path) [406, Not Acceptable]
    - Accept header is application/vnd.ccss.standardstatementcollection+XML
        - Result SHALL be the collection of all statements in the level immediately below the root specified in the path as application/vnd.ccss.standardstatementcollection+XML
    - Accept header is application/vnd.ccss.standardstatement+XML
        - Result SHALL be a Request-0102 error (Requesting an item for a collection path) [406, Not Acceptable]
    - Accept header is any other media type
        - Result SHALL be a Request-0103 error (Requesting a non statement media type) [406, Not Acceptable]

Resource Type is statement/

- Resource Name is a statement classification path with a terminal slash
  AND
- Resource URI includes a ;r (recursive subtree) parameter in the final path segment
- An included ;loc (location) parameter is an error
        - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- An included ;http (web-resolvable locator) parameter is an error
        - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]

    - Accept header is omitted
        - Result SHALL be the collection of all statements in the subtree rooted at the path (omitting the root) as application/vnd.ccss.standardstatementcollection+JSON
    - Accept header is application/vnd.ccss.standardstatementcollection+JSON
        - Result SHALL be the collection of all statements in the subtree rooted at the path (omitting the root) as application/vnd.ccss.standardstatementcollection+JSON
    - Accept header is application/vnd.ccss.standardstatement+JSON
        - Result SHALL be a Request-0102 error (Requesting an item for a collection path) [406, Not Acceptable]
    - Accept header is application/vnd.ccss.standardstatementcollection+XML
        - Result SHALL be the collection of all statements in the subtree rooted at the path (omitting the root) as application/vnd.ccss.standardstatementcollection+XML
    - Accept header is application/vnd.ccss.standardstatement+XML

    - Result SHALL be a Request-0102 error (Requesting an item for a collection path) [406, Not Acceptable]
  - Accept header is any other media type
    - Result SHALL be a Request-0103 error (Requesting a non statement media type) [406, Not Acceptable]

Resource Type is id/

- Resource Name is omitted
- An included resource URI ;r (recursive subtree) parameter is ignored (redundant)
- An included ;loc (location) parameter is an error
    - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- An included ;http (web-resolvable locator) parameter is an error
    - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- Included ;v (version) parameters is an error
    - Result SHALL be a Request-0109 error (Versioning not applicable) [406, Not Acceptable]

  - Accept header is omitted
    - Result SHALL be the collection of all statements in the statement data store as application/vnd.ccss.standardstatementcollection+JSON
  - Accept header is application/vnd.ccss.standardstatementcollection+JSON
    - Result SHALL be the collection of all statements in the statement data store as application/vnd.ccss.standardstatementcollection+JSON
  - Accept header is application/vnd.ccss.standardstatement+JSON
    - Result SHALL an error (requesting an item for a collection path)
  - Accept header is application/vnd.ccss.standardstatementcollection+XML
    - Result SHALL be the collection of all statements in the statement data store as application/vnd.ccss.standardstatementcollection+XML
  - Accept header is application/vnd.ccss.standardstatement+JSON
    - Result SHALL be a Request-0102 error (Requesting an item for a collection path) [406, Not Acceptable]
  - Accept header is any other media type
    - Result SHALL be a Request-0103 error (Requesting a non statement media type) [406, Not Acceptable]

Resource Type is id/

- Resource Name is a statement id
- An included resource URI ;r (recursive subtree) parameter is an error
    - Result SHALL be a Request-0105 error (Requesting subtree for a single item) [406, Not Acceptable]
- An included resource URI ;size (size) parameter is an error
    - Result SHALL be a Request-0106 error (Requesting paginated results for a single item) [406, Not Acceptable]
- Included ;v (version) parameters is an error
    - Result SHALL be a Request-0109 error (Versioning not applicable) [406, Not Acceptable]

  - Accept header is omitted
    - Result SHALL be the requested statement as application/vnd.ccss.standardstatement+JSON
  - Accept header is application/vnd.ccss.standardstatementcollection+JSON
    - Result SHALL be an error (requesting a collection for an item path)

- Accept header is application/vnd.ccss.standardstatement+JSON
    - Result SHALL be the requested statement as application/vnd.ccss.standardstatement+JSON
- Accept header is application/vnd.ccss.standardstatementcollection+XML
    - Result SHALL be a Request-0101 error (Requesting a collection for an item path) [406, Not Acceptable]
- Accept header is application/vnd.ccss.standardstatement+XML
    - Result SHALL be the requested statement as application/vnd.ccss.standardstatement+XML
- Accept header is any other media type
    - Result SHALL be a Request-0103 error (Requesting a non statement media type) [406, Not Acceptable]

Resource Type is vocabulary/

- Resource Name is omitted
    - Result SHALL be a Request-0007 error (Vocabulary name not specified) [400, Bad Request]

Resource Type is vocabulary/

- Resource Name is a vocabulary with or without a terminal slash
- An included resource URI ;r (recursive subtree) parameter is an error
    - Result SHALL be a Request-0105 error (Requesting subtree for a single item) [406, Not Acceptable]
- An included resource URI ;size (size) parameter is an error
    - Result SHALL be a Request-0106 error (Requesting paginated results for a single item) [406, Not Acceptable]
- An included ;loc (location) parameter is an error
    - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]
- An included ;http (web-resolvable locator) parameter is an error
    - Result SHALL be a Request-0107 error (Requesting location for an item that is not an identifier) [406, Not Acceptable]

- Accept header is omitted
    - Result SHALL be the requested vocabulary as application/vnd.ccss.vocabulary+JSON
- Accept header is application/vnd.ccss.vocabulary+JSON
    - Result SHALL be the requested vocabulary as application/vnd.ccss.vocabulary+JSON
- Accept header is application/vnd.ccss.vocabulary+XML
    - Result SHALL be the requested vocabulary as application/vnd.ccss.vocabulary+XML
- Accept header is any other media type
    - Result SHALL be a Request-0104 error (Requesting a non vocabulary media type) [406, Not Acceptable]

Annex: Resource Publication

An HTTP PUT message MAY be used to:

- publish a new statement into the statement data store
- publish a new version of a statement in the statement data store
- update an existing version of the statement in the statement data store

The following rules SHALL be used to govern the publishing/update process.

The rules SHALL take precedence over any other stated behavior in this Specification.

Taxon data in the learning standards statement to be published does not correspond with the Resource Name

- Result SHALL be a Validation-1313 error (Taxon value does not correspond to Resource Name) [400, Bad Request]

Resource Name identifies a collection and not an individual statement

- Result SHALL be a Request-0010 error (Resource Name is a collection when only a statement is allowed) [400, Bad Request]

Version parameters of the *non-terminal* path segments in the Resource Name do not specify an existing statement in the statement store

- Result SHALL be a Request-0108 error (A version request is not valid) [400, Bad Request]

Version parameter is omitted from the *terminal* path segment in the Resource Name

- Statement does not exist in the statement data store
  - Statement SHALL be added to the statement data store (version as specified in the statement) [201, Entity Created]
- Statement exists in the statement data store (including version as specified in the statement)
  - Statement duplicates existing statement
    - Result SHALL be a Request-0204 error (Attempting to publish a learning standards statement that already exists) [405, Method Not Allowed]
  - Statement is not a valid update, should be a new version
    - Result SHALL be a API-2301 error (The statement must be published as a new version, not an update) [400, Bad Request]
  - Statement is a valid update
    - Statement SHALL be updated *in place* in the statement data store [204, No Content]

Version parameter is included with the *terminal* path segment in the Resource Name

- Statement/version does not exist in the statement data store
  - Statement SHALL be added to the statement data store (version as specified in the statement and path segment) [201, Entity Created]
- Statement exists in the statement data store (including version as specified in the statement)
  - Statement duplicates existing statement
    - Result SHALL be a Request-0204 error (Attempting to publish a learning standards statement that already exists) [405, Method Not Allowed]
  - Statement is not a valid update, should be a new version
    - Result SHALL be a API-2301 error (The statement must be published as a new version, not an update) [400, Bad Request]
  - Statement is a valid update
    - Statement SHALL be updated *in place* in the statement data store [204, No Content]

Annex: Conformance Test Cases

A conforming implementation of the API SHALL pass all of the test cases listed.

- Validating message URIs (URI Base, Version, Resource Type, Resource Name and Resource Name parameters).
- Validate JSON submissions (valid JSON, schema validation, semantic validation).
- Validate all method behaviors (retrieve, publish, update, delete, metainformation).
- Validate authentication.
- Validate pagination behavior.

Testing SHALL include both valid request forms and erroneous forms.

To Do: Fully define all conformance test cases.

Annex: Browser Access URIs

The API URIs are designed to be used in a client application or through a command line interface such as CURL. They cannot be used directly in a web browser.

As shown in Figure 1, a Publishing Agent MAY deploy a web server that provides an interface between a web browser and the application server that implements the API (the same HTTPD could be used for both the web server and the application server).

The web server SHALL support HTTP scheme URIs and only the HTTP GET message. The HTTP scheme URI form supported by the web server SHALL use a subset of the full API resource URIs:

- The URI scheme SHALL be http.
- The https URI scheme SHALL NOT be used.
- The URI authority component SHALL be specified by the Publishing Agent.
- The API Resource URI *API Base URI* part (api) SHALL be omitted from the URI path component.
- The API Resource URI *API Version* part (v1) SHALL be omitted from the URI path components (the http scheme URIs only support the current version of the API).
- The Publishing Agent MAY specify an initial URI path component.
- The API Resource URI *Resource Type* part as specified herein SHALL appear in the http scheme URI as the first element following the initial Publishing Agent-specific path component.
- The API Resource URI *Resource Name* part as specified herein MAY appear in the http scheme URI following the Resource Type component.
- The ;size (size) Resource Name parameter SHALL NOT be used (all requests attempt to return the entire results set; pagination is not supported).
- The other Resource Name parameters MAY be used.
- Authentication SHALL NOT be used.
- To request the results in the JSON serialization, a &json path segment MAY be added to the end of the URI (JSON is the default results serialization, the path segment is not REQUIRED).
- To request the results in the XML serialization, an &xml path segment MAY be added to the end of the URI.

When returning an error message, the web server MAY return the JSON form of the error message. The web server SHOULD return an HTML encoded form of the error message.

To Do: Define the HTML encoded form of an error message.

Except as noted above, the URI format and message behavior SHALL be as specified herein.

The table below illustrates API URI messages and corresponding HTTP scheme URIs. The http scheme URIs do not include a Publishing Agent initial URI path component.

URI Form	URI	Notes
API	GET /api/v1/statement/CCSS/Math/Content/./8/Math/G/A/1/a HTTP/1.1	Single Statement
HTTP	http://publishingagent.tld/statement/CCSS/Math/Content/./8/Math/G/A/1/a	
API	GET /api/v1/statement/CCSS/ELA;r HTTP/1.1	Statements in XML (recursive)
HTTP	http://publishingagent.tld/statement/CCSS/ELA;r&xml	
API	GET /api/v1/id/5b37819c9b0d43eaafca4a86fb7091e7;http HTTP/1.1	ID
HTTP	http://publishingagent.tld/id/5b37819c9b0d43eaafca4a86fb7091e7	
API	GET /api/v1/id/5b37819c9b0d43eaafca4a86fb7091e7;r;v=2.5 HTTP/1.1	Versions and Recursive (error)
HTTP	http://publishingagent.tld/id/5b37819c9b0d43eaafca4a86fb7091e7;r;v=2.5	

Annex: GIM-CCSS Statement Taxon Structure

Note: This annex is informative.

The GIM-CCSS learning standards statements use the following taxon structure to classify statements.

Initiative
Framework
Set
Strand
Grade
Discipline
Domain
Cluster
Standard
Component
Granular A
Granular B

Additional granularity levels MAY be added as needed (e.g., Granular C, Granular D). A complete classification is applied to each statement

A Resource Name in a Resource URI concatenates the values of the taxons, in the order listed above, treating each as a URI path segment, e.g.,
Initiative/Framework/Set/Strand/Grade/Discipline/Domain/Cluster/Standard/Component/Granular A/Granular B

As described, each path segment MAY be augmented with a version label.

GIM CCSS Project Documentation

Scope, Technical Requirements, Approaches, and Recommendations

Learning Standards Digital Representation Specification: JSON Serialization Schemata

Learning Standards Digital Representation Specification: XML Serialization Schemata

Learning Standards Digital Representation Specification: RESTful API [this document]

Learning Standards Digital Representation Specification: Publishing Agent Design and Requirements

Change Log

Date	Version	Author	Notes
20130304	0.4	DR	Initial draft. Release to management group for review.
20130319	0.6	DR	Release to working group for review.
20130912	1.0	DR	V1.0 public release.