REST API

What is DSpace REST API

The REST API module exposes machine readable representations of content in Communities, Collections, Items, and Bitstreams.

The DSpace 4.0 REST API (Jersey) allows for data in DSpace to be re-used by external systems to make new uses of your data. The DSpace 4.0 REST API provides READ-ONLY access via JSON or XML to publicly accessibly Communities, Collections, Items and Bitstreams. Only non-hidden item metadata (e.g. provenance is hidden by default) are exposed at the Item endpoint. We intend that future DSpace releases will grow and evolve the REST API to support a greater set of features, based on community input and support. This Jersey implementation of a REST API for DSpace is not related to other add-on modules providing REST-API support for DSpace, such as GSOC REST API, Wijiti REST API, Hedtek REST API, or SimpleREST.

REST Endpoints

We have modeled the DSpace entities of Communities, Collections, Items, and Bitstreams. The API is not a straight database schema dump of these entities, but provides some wrapping that makes it easy to follow relationships in the API output.

HTTP Header: Accept

Note: You must set your request header's "Accept" property to either JSON (application/json) or XML (application/xml) depending on the format you prefer to work with.

Example usage from command line in XML format with pretty printing:

```
curl -s -H "Accept: application/xml" http://localhost:8080/rest/communities | xmllint --format -
```

For this documentation, we will assume that the URL to the "REST" webapp will be http://localhost:8080/rest/ for production systems, this address will be slightly different, such as: http://demo.dspace.org/rest/. The path to an endpoint, will go after the /rest/, such as /rest/communities, all-together this is: http://localhost:8080/rest/communities

Another thing to note is that there are Query Parameters that you can tack on to the end of an endpoint to do extra things. The most commonly used one in this API is "?expand". Instead of every API call defaulting to giving you every possible piece of information about it, it only gives a most commonly used set by default and gives the more "expensive" information when you deliberately request it. Each endpoint will provide a list of available expands in the output, but for getting started, you can start with ?expand=all, to make the endpoint provide all of its information (parent objects, metadata, child objects). You can include multiple expands, such as: ?expand=collections, subCommunities .

Communities

Communities in DSpace are used for organization and hierarchy, and are containers that hold sub-Communities and Collections. (ex: Department of Engineering)

List Communities	`/communities/`
Specific Community	`/communities/:communityID`
Community Expands	`parentCommunity, collections, subCommunities, logo, all`

Collections

Collections in DSpace are containers of Items. (ex: Engineering Faculty Publications)

List Collections	`/collections/`
Specific Collection	`/collections/:collectionID`
Collection Expands	`parentCommunityList, parentCommunity, items, license, logo, all`

You can access all the collections in a specific community through: /communities/:communityID?expand=all

Items

Items in DSpace represent a "work" and combine metadata and files, known as Bitstreams.

Specific Item	`/items/:itemID`
Item Expands	`metadata, parentCollection, parentCollectionList, parentCommunityList, bitstreams, all`

You can access all the items in a specific collection through: /collections/:collectionID?expand=items

Bitstreams

Bitstreams are files. They have a filename, size (in bytes), and a file format. Typically in DSpace, the Bitstream will the "full text" article, or some other media. Some files are the actual file that was uploaded (tagged with bundleName:ORIGINAL), others are DSpace-generated files that are derivatives or renditions, such as text-extraction, or thumbnails. You can download files/bitstreams. DSpace doesn't really limit the type of files that it takes in, so this could be PDF, JPG, audio, video, zip, or other. Also, the logo for a Collection or a Community, is also a Bitstream.

Specific Bitstream	`/bitstreams/:bitstreamID`
Download a Bitstream	`/bitstreams/:bitstreamID/retrieve`
Bitstream Expands	`parent, all`

You can access all the Bitstreams in a specific Item through: /items/:itemID?expand=bitstreams

You can access the parent object of a Bitstream (normally an Item, but possibly a Collection or Community when it is its logo) through: /bitstreams/:bitstreamID?expand=parent

Introduction to Jersey for developers

The REST API for DSpace 4.0 is implemented using Jersey, the reference implementation of the Java standard for building RESTful Web Services (JAX-RS 1, JSR 311). That means this API should be easier to expand and maintain than other API approaches, as this approach has been widely adopted in the industry.

Below is some sample Jersey code of how you wire up resources, choose to serialize to HTML, JSON or XML. And between display single-entity vs. display list-of-entities.

```
@Path("/collections")
public class CollectionsResource {
    @GET
    @Path("/")
    @Produces(MediaType.TEXT_HTML)
    public String listHTML() {...}


    @GET
    @Path("/")
    @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
    public org.dspace.rest.common.Collection[] list(@QueryParam("expand") String expand) {...}


    @GET
    @Path("/{collection_id}")
    @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
    public org.dspace.rest.common.Collection getCollection(@PathParam("collection_id") Integer collection_id,
@QueryParam("expand") String expand) {...}
```

There was no central ProviderRegistry that you have to declare your path. Your free to use @annotations to get your code to respond to requests. There are helpful parameter helpers to extract parameters into Java variables.

Configuration for DSpace REST

Property	stats
Example Value	true
Informational Note	Boolean value indicates whether statistics should be recorded for access via the REST API; Defaults to 'false'.

Recording Proxy Access by Tools

For the purpose of more accurate statistics, a web-based tool may specify who is using it, by adding parameters to the request:

```
http://localhost:8080/rest/items/:ID?userIP=ip&userAgent=userAgent&xforwarderfor=xforwarderfor
```

If no parameters are given, the details of the HTTP request's sender are used in statistics. This enables tools to record the details of their user rather than themselves.

Deploying the DSpace REST API in your Servlet Container

The dspace-rest module is automatically configured to compile and build with DSpace 4.0, so a mvn+ant process will create the webapp. To make it work in your environment, you would just need to add a context entry for it in your servlet container. For example, in tomcat, one might alter $CATALINA_HOME /conf/server.xml and add:

```
<Context path="/rest" docBase="/dspace/webapps/rest"/>
```

Additional Information

Additional information can be found in the README for dspace-rest, and in the GitHub Pull Request for DSpace REST (Jersey).