1. Contextual Metadata Extraction

According to the Research Data Alliance definition, the process of “contextual metadata extraction” consists in “creating metadata associated with files and collections[1].” In order to offer this functionality to our users and administrators, PECE installations come by default with a pre-configured Open API which allows for indexing, creating, updating, and deleting data, metadata, and provenance information about different content types.

Our Open API is composed by two servers: 1) a REST server with JSON and XML endpoints (URLs from which to interface with the platform) and a SPARQL endpoint which can be used to query RDF datasets, leveraging semantic web extensions in PECE.

Our REST service is divided into two basic endpoints: one for anonymous, non-authenticated users access to public, CC-licensed content and one for authenticated users with access to restricted content. The public endpoint has three resources: *files, nodes, and artifacts.* The restricted endpoint includes the three resources plus another one for handling user information: *users*. In the restricted endpoint, three resources support (*files, nodes, users*) support performing the following functions: Retrieve (GET), Create (POST), Update (PUT) and Delete (DELETE).

The resource *artifacts* has a filtered output for consumption by systems other than PECE/Drupal, therefore it only accepts (GET) requests. It contains all the data and metadata fields for each artifact archived in your PECE instance and can be accessed by displaying the name of the resource you want to consume: *image_artifacts, video_artifacts, audio_artifacts, text_artifacts, webpage_artifacts, bundle_artifacts*.

Here is the table with the relation of supported functions for each endpoint and resource:

RESOURCE	ENDPOINTS							
	api/public				api/auth			
	GET	POST	PUT	DELETE	GET	POST	PUT	DELETE
Files	X				X	X	X	X
Nodes	X				X	X	X	X
Users					X	X	X	X
Artifacts	X				X			

Public and CC-licensed content will be accessible both the “public” and “auth” endpoints, whereas restricted content can only be accessed through “auth” (private) endpoint. Authorized users and administrators have much more flexibility as they can create, modify, and delete content for which they have permission to do so. This flexibility includes the ability to manipulate user accounts and content in batches.

1 Source: http://smw-rda.esc.rzg.mpg.de/index.php/Contextual_metadata_extraction

Next we will describe how to obtain structured and serialized data from the public interface, then we will describe how to use the Open API to alter content, which is extremely useful for the purposes of data migration (and syncing across data repositories and web applications).

PECE Open API can be accessed through the following URLs (changing the portion with your respective domain name):

```
// For public, anonymous users:
https://your-domain.org/api/public/files
https://your-domain.org/api/public/nodes
https://your-domain.org/api/public/image_artifact, video_artifact, and so on.

//For authenticated users:
https://your-domain.org/api/auth/files
https://your-domain.org/api/auth/nodes
https://your-domain.org/api/auth/nodes
https://your-domain.org/api/auth/image_artifact, video_artifact, and so on.
```

Responses can be formatted either in XML (Extensible Markup Language) and JSON (JavaScript Object Notation), “application/xml” (default) and “application/json” respectively.

Suppose you want to request machine-readable data and metadata from your PECE instance. The following command would return a JSON document with all the data and metadata fields for a particular node whereas “nid” is the “Node Identifier Number”:

```
$ curl -X GET https://your-domain.org/api/public/nodes/nid.json
```

The following output would be the result, exposing data and metadata for the requested node:

```
{
    "changed": "1439121431",
    "comment": "1",
    "comment_count": "0",
    "created": "1439121000",
    "field_collaborators": [],
    "field_critical_commentary":,
    "field_group_audience":,
    "field_format":
    "field_image_annotation": [],
    "field_licensecc":,
    "field_location":,
    [...]
}
```

To render the previous output in XML, the syntax of your request would be the similar, except that the termination (.json) would have to be modified (or omitted) as in the example below:

```
$ curl -X GET https://your-domain.org/api/public/nodes/nid.xml
```

If you want to retrieve index lists of nodes or files, you just have to omit the last portion of the URL with “Node ID”. Please note that the GET function only lists 20 items by default. If you need retrieve more (or less) items, it necessary to pass a parameter in the URL.

For the purposes of interoperability with other web frameworks and data repositories, we created filtered XML and JSON outputs for each PECE content type (with permissions to publicly accessible). Filtered outputs were specified to be both machine-readable and comprehensible by humans. In order to obtain, for instance, a listing of “image artifacts,” the following commands could be executed:

```
// For the complete listing in XML:
$ curl -X GET https://your-domain.org/api/public/image_artifacts

// For the complete listing in JSON:
$ curl -X GET -H "Accept: application/json" \
       https://your-domain.org/api/public/image_artifacts
```

Please note the filtered output follows the convention of the PECE Data Model (in appendix). Consult this document to understand the data types and the relationships between fields:


```
{
    "URI": "F3EA8139A6B43ECBC56BB7CF51E51",
    "Title": "Orion Nebula",
    "Date of Creation": "1439121000",
    "Revision Number": "23",
    "Author": "John Public",
    "Collaborators": "Alice S.",
    "Format": "JPEG",
    "Project": {
        "Name": "Minority Astronomers Multi-Disciplinary Collaborations",
        "Description": "This project investigates how women scientists engaged
in collaborative, multidisciplinary research build relationships and the
effects of these relationships on their careers [...]",
        "Members": "Bob M., Alice S., John Public, Mary B.",
        "Funding Agency": "NSF EAGER"
    },
    "Fieldsites": "Astroinformatics",
    "Annotations": [],
    "Commentary": "Image captured by the \"ACS\". According to the Hubblesite,
more than 3,000 stars of various sizes appear in this image.",
    "License": "//creativecommons.org/licenses/by/3.0/",
    "Tags": "NASA, Hubble, astroinformatics, Creative Commons",
    "Image URL": "https://astroanthro.net/public/nebula.jpg",
    "Location": {
        "lat": "20",
        "lat_cos": "0.93969262078591",
        "lat_sin": "0.34202014332567",
```

```
        "lng": "-20",
        "lng_rad": "-0.34906585039887"
    },
    "Group audience": "NSF/EAGER Astroinformatics research group"
    [...]
}
```

In the example above, we have information on a particular artifact with provenance fields such as “project” and “fieldsite” – with relational information about project the field in which the data was produced by a team of ethnographers – and relational fields, such as “group audience,” “collaborators,” (which lists ethnographers who contributed content, but are not the “authors” of a particular piece of data) and “annotation” (which lists all the annotations that were generated by one or multiple users of the platform).

For complete data manipulation capabilities through the “auth” endpoint, it is necessary to have an account in the platform (as well as permission to manipulate the content you are requesting). If you are a registered PECE user identified with a “researcher” role, you are granted control over the content you generated, including the possibility to create, modify, retrieve, and delete content or specific fields of particular types of content. Administrators are the recommended users to perform most tasks through the “auth” endpoint. For security purposes, we can restrict access to the “auth” endpoint only to users or unable it entirely (or grant access to it only to certain machines, see the section on PECE Security for further information on access control).

Let's suppose that, at some point, the necessity to update a particularly field has appeared. It became necessary for you or another member of the research team to change the “critical commentary” to include further critical evaluation of a particular artifact. This command would accomplish this task by changing content of the field “critical commentary”:

```
$ curl -X PUT -H "Content-Type: application/json" \
       -H "Cookie: EXAMPLE_SESS02caabc123=ShBy6ue5TTabcdefg" \
       -H "X-CSRF-Token: EXAMPLE_w98sdb9udjiskdjs" \
       -H "Accept: application/json" \
       -d '{"nid":"18", "field_critical_commentary":"New Kritik"}' \
       https://your-domain.org/api/auth/nodes/18
```

As you can tell from the example above, there many parameters to pass to `curl` when creating, deleting, or modifying a node, file, or user. First, it is necessary to log-in to the platform through the “users” resource:

```
$ curl -X POST -H "Content-Type: application/json" \
       https://astroanthro.net/api/auth/users/login.json \
       -d '{"username":"user","password":"password"}' \
       -c session.txt
```

Since we are using the restricted “auth” endpoint, please observe that it fundamental to collect and then pass the information about your X-CSRF (cross-site request forgery) token and session information (“cookie”) as header parameters in every subsequent request. This can be accomplished in many ways. For instance, you can save it to a text file with the `-c` parameter with `curl` then execute every POST or PUT request passing the `-b` parameter plus the name of the file you created:

```
$ curl -X GET -H "Content-Type: application/json" \
       https://your-domain.org/api/auth/users/nid.json \
       -b session.txt
```

The command above would provide the information on a particular user. A similar syntax applies for requesting other types of data. Please observe that it is necessary to pass the parameter on Node ID (“nid”) or User ID (“uid”) if you are accessing, modifying, or deleting a resource. Your request must also include the body data (which is identified in the platform by the machine name of the field you want to modify – please consult the document “PECE Data Model” for the description of mappings from “field_machine_name” to “field name”).

There are many benefits in using the Open API for administrative tasks. It is possible to perform tasks in bulk, modifying large swaths of data in batches. It is also useful to modify punctually and quickly any type of data, including artifacts, files, and users. For the purposes of promoting Open Data exchange and Open Access among ethnographers more generally, our API allows for automated tasks of contextual metadata extraction as well as data harvesting. The technical details regarding Open Data exchange are further discussed under the section “Disposition” of this document.

Instruction for developers: This data management functionality depends on the Drupal 7.x module “Services, version 3.12” which comes pre-installed and configured with PECE 1.0. Our API also depends on the following extra modules: “views”, “ctools”, and “libraries”. When using this tutorial for testing purposes, make sure curl is compiled against OpenSSL with support for HTTPS. This can be verified by asked for the version of `curl` with the parameter `-v` at the command line. In the appendix, we have a copy of the exported files configuration for each PECE Open API “endpoint” and “resource” described in this section.