NEES Web Services 2.0

1. Overview

This document provides information about the NEES Web Services application programming interface (API) and associated data formats.

2. REST-style Communication

The NEES Web Services make use of representational state transfer (REST)-style communication between clients and the server. Communication is initiated by the client in the form of HTTP GET, DELETE, POST, and PUT requests. GET is used to read data, DELETE is used to delete data, POST is used to create new data, and PUT is used to modify existing data. The server responds to requests by returning status codes, data, or both. Because REST transactions are stateless, each HTTP request must be accompanied by a URI that uniquely identifies the specific resource being requested. More information about valid NEES Web Services URIs is presented in section 2.1. In addition to a URI, POST and PUT requests must be accompanied by an XML document containing data that conforms to the NEES Web Services API, described in section 4.

Different HTTP status codes (and data) are returned by the server in response to requests:
GET - A successful GET returns status code 200 (OK) and XML data in the body.
PUT - A successful PUT returns status code 200 (OK) and XML data in the body.
POST - A successful POST returns status code 201 (created) and a Location header containing the URI of the newly created resource.
DELETE - A successful DELETE returns status code 204 only (no data).

Error status codes include:

400: Bad request - The request could not be understood by the server due to malformed syntax.
401: Unauthorized - Invalid username and/or password, or insufficient privileges for the command.
404: Not found - Couldn't find the resource specified by the URI.
410: Gone - The requested resource is no longer available at the server and no forwarding address is known.

2.1 URIs

Note: *This section describes the URIs used to access all data in NEES except data related to user-uploaded files and the files themselves. Access to file-related data is described in section 2.2.*

In general, a valid NEES Web Services URI begins with https://ws.nees.org:9443/REST/ as the base, followed by additional information describing the logical path to a specific resource. For example, the following URI is used to access trial three within experiment five within project

27:https://ws.nees.org:9443/REST/Project/27/Experiment/5/Trial/3

The box below depicts the NEES Web Services URI hierarchy. Complete URIs can be constructed by appending successive elements and their ID numbers to the base URI, similar to the example given above.

```
Project
 Experiment
  Organization
  CoordinateSpace
   CoordinateSpaceDataFile
  Trial
   DAQConfig
    DAQChannel
```

```
          DAQChannelEquipment
        ControllerConfig
          ControllerChannel
            ControllerChannelEquipment
        Repetition
      SensorLocationPlan
        SensorLocation
      SensorPool
      Material
      SimilitudeLawValue
    Simulation
      ExperimentModel
      SimulationRun
SimulationType
ExperimentModelType
Material
  MaterialDataFile
MaterialType
  MaterialTypeProperty
SensorManifest
CoordinateDimension
CoordinateSystem
MeasurementUnitConversion
MeasurementUnitCategory
  MeasurementUnit
EquipmentClass
  EquipmentModel
    EquipmentClassAttribute
Facility
  Equipment
    EquipmentDocumentation
    EquipmentAttributeValue
EquipmentAttribute
Attribute
DocumentFormat
ExperimentDomain
SimilitudeLawGroup
  SimilitudeLaw
SensorModel
  Sensor
    Calibration
```

NEES Web Services URI Hierarchy

2.2. File-related URIs

Path information about existing files can be obtained by looking at the <DataFile>element(s) in the XML returned by successful GET requests. For example, if a successful

```
GET https://ws.nees.org:9443/Project/4
```

request is issued, the XML returned by the server might include a snippet that looks like

```
<DataFile link="/REST/DataFile/354"/>
```

Issuing a subsequent GET request to

```
https://ws.nees.org:9443/REST/DataFile/354
```

will return a <DataFile> element for its metadata.You can locate any existing file by repeating this process.

If <DataFile> is a real file, then the <DataFile> element will contain a URL for downloading.

3. Using the NEES Web Services

3.1. curl

curl is a command line utility that transfers data to or from a server. It can be used to issue HTTP DELETE, GET, POST, and PUT requests, thus making it possible to interact with NEEScentral via the Web Services. The examples given below show how to use curl to call the Web Services.

3.2. Authentication

Before being allowed to access NEEScentral resources, you must be authenticated through a GridAuth GAsession ID. HTTP and cookie authentication are not explained in this guide.
To get a GAsession ID, first visit

```
https://central.nees.org/cgi-bin/gridauth.cgi?username=yourusername&password=yourpassword
```

and log in. Next, look for the session id as below in the return result:

```
c0bc5b115fd47455b780ec66493fa90dc96398f1
```

Copy the GAsession ID for use with curl calls. When making curl calls, append ?GAsession=<*id*> to the end of the HTTP request URI.

```
curl -k --request GET
https://ws.nees.org:9443/REST/Project?GAsession=c0bc5b115fd47455b780ec66493fa90dc96398f1
```

3.3. Getting Existing Information

```
curl -k --request GET https://ws.nees.org:9443/REST/Project?GAsession=foo
```

This command returns an XML document that lists each of the associated projects.

```
<?xml version="1.0" encoding="UTF-8"?>
<central>
  <Project link="/REST/Project/267" id="267"/>
  <Project link="/REST/Project/191" id="191"/>
  <Project link="/REST/Project/125" id="125"/>
  <Project link="/REST/Project/222" id="222"/>
  <Project link="/REST/Project/223" id="223"/>
  <Project link="/REST/Project/124" id="124"/>
</central>
```

Following the URIs given by link attributes allows you to drill down into the depths of a project.

```
curl -k --request GET https://ws.nees.org:9443/REST/Project/222?GAsession=foo
```

This returns a new XML document with further information about project 222.

3.4. Creating New Information and Modifying Existing Information

To upload new information to NEEScentral, create an XML document that conforms to the NEES Web Services API and upload it using an HTTP POST request.

```
curl -k --data-binary @inputFile.xml --request POST https://ws.nees.org:9443/REST/Project?GAsession=foo
```

Similarly, use an HTTP PUT request to modify existing information.

```
curl -k --data-binary @inputFile.xml --request PUT https://ws.nees.org:9443/REST/Project?GAsession=foo
```

3.5. Deleting Information

Use an HTTP DELETE request to delete existing information.

```
curl -k --request DELETE https://ws.nees.org:9443/REST/Project/222?GAsession=foo
```

3.6. Upload New DataFile

To upload a new file, you can use a HTTP POST to submit multipart form data to the following URL:

```
https://ws.nees.org:9443/REST/DataFile
```

The submitted multipart form data should include an uploaded file, a *GAsession* and a*message*. Here the message should be an XML document for DataFile.

```
curl -v -X POST -F upload=@/tmp/test.txt -F
"message=%3CDataFile%20viewable%3D%22MEMBERS%22%3E%3Cname%3EMy%20Test%20Data%20File1%3C/n
ame%3E%3Cpath%3E/nees/home/NEES-2007-0354.groups/Experiment-1/Trial-1/Rep-
1/Converted_Data%3C/path%3E%3CcurationStatus%3EUncurated%3C/curationStatus%3E%3C/DataFile%3E" -F
GAsession=foo https://ws.nees.org:9443/REST/DataFile/
```

where the message is the URL encoding of the following XML document:

My Test Data File again
/nees/home/NEES-2007-0354.groups/Experiment-1/Trial-1/Rep-1/Converted_DataUncurated

4. NEES Web Services API

The NEES Web Services API is located at https://ws.nees.org:9443/nees.html