Building the National Virtual Collaboratory for Earthquake Engineering Research

NEESgrid

Technical Report NEESgrid-2002-05
www.neesgrid.org

(Whitepaper Version: 1.0
Last modified: September 26, 2002)

The NEESgrid Metadata Service API: Overview

Joe Futrelle[1] Jeff Gaynor[1]

[1] National Center for Supercomputing Applications, Urbana-Champaign, IL 61820

Feedback on this document should be directed to futrelle@ncsa.uiuc.edu

Acknowledgment: This work was supported primarily by the George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES) Program of the National Science Foundation under Award Number CMS-0117853.

The NEESgrid Metadata Service API: Overview

Draft whitepaper by Joe Futrelle and Jeff Gaynor. Version: 1.0. Last modified: September 25, 2002

1. Architecture

The NEESgrid Metadata Service provides access to metadata about a variety of entities, such as experiments, researchers, apparatus, events, and facilities. The service is designed to allow remote clients to browse, update, and otherwise manage metadata objects representing these entities of interest. The metadata objects are represented in a simple, consistent manner, which allows them not only to contain attributes of various types, but also to make reference to other objects. The objects are also tracked through a versioning system and protected through access control and object-level locking. Using the metadata service, remote clients can retrieve full or partial information about objects, request uploads and downloads of data files to and from the repository, and make full or partial updates to objects.

The Metadata Service will run on the NEES POP at every site, in addition to a central Metadata Service that will run at a central site. This document describes the capabilities of the Metadata Service running on every site's NEES POP, rather than the metadata service at the central repository. The capabilities are described in terms of the client API. The client API is an implementation of a generic object access API that can be interfaced to arbitrary back-ends. In the server-side implementation of the object access API, the back-end is the repository's back-end RDBMS. In the client-side implementation, the back-end is a network protocol based on HTTPS and XML which is used to communicate with the Metadata Service. After its initial implementation, this protocol, which resembles SOAP, will be able to be extended to become a SOAP-based service which other clients can use. Initially, however, the only client that will use the client API is a set of CHEF teamlets that will provide a user interface for data and metadata management, and later for experiment management. Thus, the "client" is really a CHEF server whose clients are end users. For the remainder of this document, "client" should be understood to mean the CHEF server, except where otherwise indicated.

2. Client API

The client API is a Java class library providing remote access to the Metadata Service. The API enables clients to do several things:

1. Authenticate on behalf of a user
2. Request full or partial sets of attributes of metadata objects
3. Retrieve multiple versions of the same object
4. Resolve inter-object references
5. Update objects
6. Request upload or download of data files

The following sections describe these capabilities.

Authenticate on behalf of a user

At the beginning each request, the client will identify the end user to the metadata service using a Distinguished Name. The authenticity of the DN will be assumed as part of a trust relationship between the client and the Metadata Service, which is founded on the requirement that end users authenticate to CHEF before any client methods can be invoked. The protocol will be able to be extended to require the client to send Grid credentials. The HTTPS connection used to transport protocol messages ensures that the DN (and eventually, credentials) that the client sends will not be able to be intercepted.

The Metadata Service will enforce access control for every request based on the DN that it receives from the client. If access is forbidden, a response to that effect will be sent back, and that kind of access to the object will be denied to the client.

Request full or partial sets of attributes of metadata objects

The primary function of the Metadata Service is to provide access to metadata objects. For the client, this means that metadata objects can be partially or fully retrieved and used as if they were locally present. Since copying every object to the client is bandwidth-costly, the client will be able to request subsets of the set of metadata objects managed by the Metadata Service, and also will be able to request partial information about objects of interest.

Each object is represented in the repository as a collection of attributes of various types. Some attributes, such as the object's unique identifier, its type, its access control and its version information, are special system attributes that every object has. Other attributes represent information about real-world objects and thus belong to only to objects of certain types. For instance, an object representing a camera might have an attribute identifying its manufacturer, whereas an object representing a person would not.

The client will be able to request some or all of the attributes for a given object. This feature allows for lower-bandwidth requests when, for instance, a large collection of objects is going to be summarized for the user.

Retrieve multiple versions of the same object

Every object in the repository is associated with versioning information that identifies the sequence and timing of versions of the object. When an object is updated, the old version of the object is retained and the new version, with its time of creation (i.e. when the update was received by the Metadata Service), creator, and version number, is linked to it. The client can ask for references to all versions of an object, so that it can find out which version existed at any given time. At any time, the client can request the most recent version of any object of interest.

Resolve inter-object references

Inter-object references allow objects to use other objects as the values of attributes. For instance, an object representing an experiment may have an attribute describing the principal investigator, the value of which is an object representing a person. The client can follow these references because it can find out the ID of any object another object refers to.

Versioning complicates this considerably. If an object α created at time t_1 refers to an object β that existed at time t_1 but was updated at t_2, the reference is ambiguous. In this case, the reference from α to the original β (t_1) is retained when β is updated, but once the client follows the link from α to β (t_1), it can retrieve any version of β that meets the user's needs (next, most recent, original, etc.).

Update objects

Objects are immutable. Updating an object means creating a new version of it with modified attributes. When the client updates an object, the Metadata Service makes the necessary changes in the database and responds with a message identifying the newly-created object and its associated version information. Making changes takes time. A client can change sets of objects atomically by locking the objects, as when multiple objects that refer to each other need to be modified "at once". This prevents other clients from creating new versions of the objects, which helps prevent inconsistencies from appearing in the database. When the client releases the lock, the objects will all appear to be modified simultaneously. If the client fails to release a lock (due to failure, for instance), the lock will expire, and the affected objects will revert to their previous versions. When a client locks an object or set of objects, it asks for an expiration time and the server responds with an expiration time that is no longer than that time, or with a message that the object or any member of the set of objects are already locked.

Deleting an object in effect creates a special, final version of the object which is marked as deleted. An object can be rolled back to an earlier version, which in effect creates a new version which is in its attribute values identical to the earlier version. Undeletion user environment aliases

Root object

Request upload or download of data files

The client can request to upload or download data files. Data files are represented by objects just like everything else. However, file upload and download is handled somewhat differently from object viewing and updating, since it requires that end users have GridFTP access to the site local storage systems. For upload, the client creates or updates a file object and the Metadata Service makes the necessary changes, and additionally generates a GridFTP URL to which the client can upload the data. The client then notifies the metadata service that it is done uploading the file. The URL is temporary; further updates to the file will require additional upload requests. Downloading is accomplished similarly. The client requests a file for download, and the Metadata Service returns a URL at which it can be temporarily found. The client is then responsible for downloading the data and notifying the server that it has been downloaded.

Notification of successful upload and download will be accomplished using atomic file-level operations by the GridFTP client. When it is done either downloading or uploading a file, it will change the file's name according to a simple naming convention, and the Metadata Service will notice the name change by periodically polling the target directory.