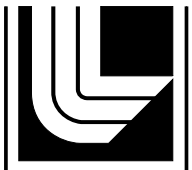

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-635754

Lorenz APIs and REST Services

J. W. Long

April 25, 2013

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Lorenz APIs and LORA/REST Services

Project Overview

Lorenz is a project aimed at using modern web technologies to make High Performance Computing (HPC) easier. It has been under development at Lawrence Livermore National Laboratory since the spring of 2010.

Lorenz is tackling three broad areas: a system "dashboard" aimed at making information about the computing center easy to determine "at a glance", job management (submission and monitoring of batch jobs), and application portals which are loosely coupled pre- and post-processors to existing codes.

API Overview

Lorenz has numerous APIs and libraries focused on different use cases.

API	Language	Used By	Purpose
Lorenz::Host Lorenz::User Lorenz::Group Lorenz::*	Perl	Lorenz Developer	Individual Perl modules for accessing specific Lorenz functionality. These modules return Perl native data structures rather than the JSON envelop returned by the REST equivalents. This should be used by developers who are implementing Lorenz internals.
Lorenz::REST::*	Perl	Lorenz Developer	The specific Perl modules invoked by the REST handler, these primary interact with the lower level Lorenz::Host|User|... modules, but provide additional structure such as a return envelope showing output, error, and status. Data returned is in JSON format.
Lorenz::Simple	Perl	Tool Developers	Aggregated Perl module for tool developers which makes it simple to access a variety of

		(cluster tools)	Lorenz functionality with minimal effort. This should be the default API to use for tools which will run on the clusters.
Lorenz::API	Perl	Tool Developers (remote tools)	An LWP-based Perl module which provides a high level API to the LORA web services. This is for developers creating applications to be run on remote hosts, although it can be used on the clusters. Requires authentication since the library acts as a browser equivalent.
Lorenz.js	JavaScript	Web App Developers (LC-hosted apps)	A high-level library which provides access to the LORA REST services to web applications. The calling sequence and function names match the Lorenz::API Perl module.
Lorenz-JSONP.js	JavaScript	Web App Developers (remote-hosted apps)	A high-level library which provides access to the LORA REST services to web applications. The calling sequence and function names match the Lorenz::API Perl module. This version uses JSONP to deal with cross-site scripting issues. (In progress.)

Lorenz::Simple – Easy-to-use Perl Module for Tool Developers on LC Clusters

The Simple.pm perl module makes it easy for tool developers on the clusters to access the range of Lorenz functionality without having to know how the methods are organized. Data are returned in perl 'native' formats rather than the JSON format returned by the REST versions of these methods.

Lorenz::Simple module for use on clusters

```
BEGIN {
    require '/usr/global/tools/lorenz/lib/lorenz.pl';
}
use strict;
use Lorenz::Simple;

my $lorenz = Lorenz::Simple->new();

my @hosts = $lorenz->getAllHosts();
my @groups = $lorenz->getUserGroups();
```

The Lorenz::Simple module provides access to all of the methods available through the individual top-level perl modules, shown in the following section.

Lorenz::* – Perl Modules for Use on Clusters

These category-specific perl modules are located in the `server/lib/perl/Lorenz` directory and are for the development of tools within the Lorenz suite as well as applications which simply make use of some Lorenz service. These modules return "native" data rather than the "wrapped" data returned by the RESTful API by default. For instance, the Lorenz::User::getUserGroups() method returns an array of group names. These modules use OO features of the language.

Conventional modules for use on clusters

```
BEGIN {
        require '/usr/global/tools/lorenz/lib/lorenz.pl';
}
use strict;
use Lorenz::Host;
use Lorenz::User;

my @hosts = Lorenz::Host->getAllHosts();
my @groups = Lorenz::User->getUserGroups();
```

Module	Methods
Lorenz::Bank	getAllBanks getBankCpuUsage getBankInfo getBankMembers getUserAccountBankInfo getUserBankDetails getUserBankObjs getUserBanks
Lorenz::Cluster	getAllClusterDetails getAllClusters getClusterBackfill getClusterBatchDetails getClusterBatchStat us getClusterDetails getClusterInfo getClusterJobLimits getLoginNodeStatus getMachStatus getUserClusters getUserProcessesByCluster
Lorenz::Host	clustername getAllHosts getAvailableLoginNodes getDeadHosts getHostSpecs getHostStatus getLoginNodes isStorageHost pickLoginNode
Lorenz::System	getDirectoryList killProcesses runApprovedCommand runCommand runCommand2 runCommandWithOpts runLorenzCommand safeRun
Lorenz::User	clearUserLorenzTmpDir debugLog getAllGroups getAllUserInfo getGroupInfo getGroupMembers getOun getOunFromUsername getUserContacteesByOun getUserDefaultHost getUserEmail getUserEnclaveStatus getUserFileTransferHostsInfo getUserGroups getUserHomeDir getUserHomeDirOnHost getUserHosts getUserInfo getUserJumpdirs getUserLorenzDir getUserLorenzTmpDir getUserSshHosts getUsername getUsernameFromOun listAllUsers lookUpCoord
Lorenz::Util	callersId expandAbbrevs fileModifiedWithin getCanonicalPath getChildren getLastModTime getWeather isApprovedHost isApprovedPath isValidFileName lorenz_from_json moveFiles sendEmail sendEmail_WithMail sendEmail_WithSMTP sendEmail_WithSendmail timeoutExec toText wraperr wrapit wrapjson wrapout

Lorenz::API - Lorenz Perl Module for Remote Use

For remote applications, or for use on the clusters when Kerberos credentials are not available, an LWP-based module called Lorenz::API is available. Because the LC web server requires an OTP authentication, you must provide authentication information.

Lorenz::API – Perl module for use on remote hosts

```
use strict;
use Lorenz::API;

# For command line tools, you can have the library prompt you for username and
password
my $api = Lorenz::API->new( { prompt=>1 } );

# ...or you can provide them directly
my $api = Lorenz::API->new( { username=$username, password=$password } );

my $response = $api->getAllHosts();

# response is a standard HTTP::Response object
if ($response->is_success) {
      print $response->content;
} else {
      print $response->status_line, "\n";
}
```

Brief Summary of Lorenz::API methods

Lorenz::API methods	cancelJob editJobParams execCommand executeRoute getAccounts getAllBanks getAllClusterUtilizations getAllMachineLoads getAllUserInfo getAllUserProcesses getBankHistory getBankHistoryForHost getBankInfo getBankMembership getBanks getClusterBatchDetails getClusters getCompletedJobs getCpuUsage getDefaultHost getDiskQuotaInfo getEnclaveStatus getFile getFileSystemStatus getGivetake getGroupInfo getGroups getHostDetails getHostInfo getHostJobLimits getHostTopology getJobDetails getJobs getJobsForHost getLoginNodeStatus getMachineStatus getMyJobs getNetworkInfo getNews getNewsItem getPathStat getPurgedFiles getSSHhosts getScratchFilesystems getTransferHosts getUltraCurve getUltraCurveListing getUserBanksByHost getUserInfo getUserOun getUserProcessesByCluster giveFiles holdJob killProcess new readFile sendJobSignal storeAppend storeDelete storeDelete storeRead storeWrite submitJob tailFile takeFiles unholdJob

The Lorenz::API module provides an interface that is nearly identical to the Lorenz.js JavaScript library. There also is a lower-level method, executeRoute(), which lets you specify the route explicitly. E.g.,

```
my $response = $api->executeRoute("/user/smith/groups", "get", {} );
```

Extended Summary of Lorenz:API and Lorenz.js libraries

These two libraries are for remote applications. The Lorenz.js library can be loaded in a remote web application with:

```
<script type="text/javascript"
src="https://lc.llnl.gov/lorenz/js/objects/Lorenz.js"></script>
```

All perl methods take an optional final argument, $args, a hashref containing additional arguments to the low-level endpoints.

All JavaScript methods take an option uOptions argument containing AJAX options.

Method[1]	REST Endpoint	Description
cancelJob($host,$jobid)	/queue/:host/:jobid [delete]	Cancels the given job.
editJobParams($host,$jobid)	/queue/:host/:jobid [put]	Modifies the parameters for an idle (non-running) job.
execCommand($host,$command)	/command/:host [post]	Executes the given command and returns stdout, stderr, and exit status from it.
getAccounts($user)	/user/:user/hosts [get]	Returns list of hosts on which the user has an account.
getAllBanks()	/banks [get]	Returns list of all SLURM/MOAB banks.
getAllClusterUtilizations()	/status/clusters/utilization/hourly2 [get]	Returns hour-by-hour cluster utilization for the past 14 days for each cluster.
getAllMachineLoads()	/status/clusters [get]	Returns slurm status for each cluster,

		including idle, allocated, and total nodes per partition.
getAllUserInfo()	/users [get]	Returns a list of all users along with additional information about each.
getAllUserProcesses()	/user/ME/cluster/processes [get]	Get list of the calling user's processes on all hosts.
getBankHistory($bank)	/bank/:bank/cpuutil/daily [get]	Returns daily bank utilization for given bank across the center.
getBankHistoryForHost($bank,$host)	/cluster/:host/bank/:bank/cpuutil/daily [get]	Returns daily bank utilization for given bank on one host.
getBankInfo($user,$bank)	/user/:user/bank/:bank [get]	Returns information about given bank for given user.
getBankMembership($host,$bank)	/bank/:bank/membership/:host [get]	Returns members of a given bank on the given host.
getBanks($user)	/user/:user/banks [get]	Returns list of SLURM/MOAB banks to which user belongs.
getClusterBatchDetails($host)	/user/ME/cluster/:host/batchdetails [get]	Returns job-submission details for each partition on given host.
getClusters($user)	/user/:user/clusters [get]	Returns list of hosts on which

		given user can submit a job.
getCompletedJobs($user,$period)	/user/:user/queue [get]	Returns details about jobs completed by given user over the last $period days.
getCpuUsage($user)	/user/:user/cpuutil/daily [get]	Returns day-by-day cpu usage for each cluster.
getDefaultHost($user)	/user/:user/default/host [get]	Returns the default host for Lorenz activities requiring an LC host.
getDiskQuotaInfo($user)	/user/:user/quotas [get]	Returns user's disk quota information for home directory and other filesystems.
getEnclaveStatus($user)	/user/:user/enclavestatus [get]	Returns details about the given user's HPC enclave status.
getFile($host,$path)	/file/:host/:path [get]	Retrieves the given file.
getFileSystemStatus()	/status/filesystem [get]	Returns capacity, type, and percent used for all filesystems.
getGivetake()	/user/ME/givetake [get]	Returns give/take status for the active user.
getGroupInfo($group)	/:group [get]	Returns information such as GID,

		group membership, approver for given group.
getGroups($user)	/user/:user/groups [get]	Returns list of LC groups to which user belongs.
getHostDetails($host)	/cluster/:host/details [get]	Returns key value pairs about given host, including configuration information, hardware specs, filesystem mounts, etc.
getHostInfo($host)	/host/:host [get]	Returns information about submitting jobs on given host.
getHostJobLimits($host)	/cluster/:host/joblimits [get]	Returns information about submitting jobs on given host.
getHostTopology($host)	/cluster/:host/topo [get]	Returns an image showing hardware topology for given host.
getJobDetails($host,$jobid)	/queue/:host/:jobid [get]	Returns live details about the given job.
getJobs($user)	/user/:user/queue [get]	Returns the given user's current job queue across the center.
getJobsForHost($user,$host)	/user/:user/queue [get]	Returns the given user's current job

		queue on the given cluster.
getLoginNodeStatus()	/status/loginNode [get]	Get up/down and other stats for each login node on each cluster.
getMachineStatus()	/status/machines [get]	Returns the up/down status and number of users on each cluster.
getMyJobs()	/user/ME/queue [get]	Returns the calling user's current job queue across the center.
getNetworkInfo()	/support/network [get]	Returns the network indicator.
getNews()	/news [get]	Returns list of all news items.
getNewsItem($item)	/news/:item [get]	Returns an individual news item.
getPathStat($host,$path)	/file/:host/:path [get]	Returns stat() information about the given path.
getPurgedFiles($user)	/user/:user/purgedFiles [get]	Returns the list of files the given user has had purged from Lustre.
getSSHhosts($user)	/user/:user/sshhosts [get]	Returns a list of hosts to which the given user can ssh.
getScratchFilesystems()	/scratchfs [get]	Returns a list of the scratch file systems.

getTransferHosts($user)	/user/:user/transferhosts [get]	Returns a list of hosts to which the user can transfer files.
getUltraCurve($host,$path,$curveid,$start)	/data/:host [post]	Retrieve a specific ultra curve from the given file.
getUltraCurveListing($host,$path)	/data/:host [post]	Returns a list of curves in the given ultra file.
getUserBanksByHost($user)	/user/:user/bankhosts [get]	Returns list of SLURM/MOAB banks for user, arranged by host.
getUserInfo($user)	/user/:user [get]	Returns details about the given user.
getUserOun($user)	/user/:user/oun [get]	Returns the given user's OUN and LC username.
getUserProcessesByCluster($host)	/user/ME/cluster/:host/processes [get]	Get list of the calling user's processes on the given host.
giveFiles($files,$to,$force)	/user/ME/give [post]	Give $files to the user $to, who can then use take to retrieve.
holdJob($host,$jobid)	/queue/:host/jobid [put]	Holds an idle (non-running) job.
killProcess($hosts,$pids)	/cluster/processes [post]	Kill the given processes.
readFile($host,$path)	/file/:host/:path [get]	Reads the given file (same as getFile).

sendJobSignal($host,$jobid,$signal)	/queue/:host/:jobid [put]	Sends the given job the given signal.
storeAppend($store)	/store/:store [post]	Appends to an existing information store for the calling user.
storeDelete($store)	/store/:store [delete]	Deletes the given information store for the calling user.
storeRead($store)	/store/:store [get]	Reads the given information store for the calling user.
storeWrite($store)	/store/:store [put]	Write data to the calling user's information store.
submitJob($host)	/queue/:host [post]	Submits a job to the given host.
tailFile($host,$path,$tail)	/data/:host [get]	Returns the last $tail lines of the given file.
takeFiles($target,$from,$force)	/user/ME/take [post]	Take files that were previously given by the user $from, and put them into $target.
unholdJob($host,$jobid)	/queue/:host/jobid [put]	Removes the hold for a held job.

[1] - Same method name for Lorenz::API perl modules as for Lorenz.js JavaScript library.

LORA – Lorenz RESTful API

The Lorenz RESTful API, or LORA, was modeled after the NERSC Web Toolkit, or NEWT, developed at Lawrence Berkeley National Lab. NEWT is a web service that allows you to access computing resources at NERSC through a simple RESTful API. After discussions with our colleagues at NERSC we decided to adopt their API specification and re-implement the API within our own LC computing center.

By default the LORA REST endpoints return a JSON-wrapped envelope, with output, error, and status fields. This can be controlled by adding the following parameters: `_envelope=0` (disables the envelope), `_format=raw|json` (controls whether raw output is first converted to json), `_printAndExit=1` (prints the output and then exits immediately.)

These endpoints can be accessed directly from any language that supports direct web interactions (Python, Perl, C, Java, ...), but using the Lorenz-supplied language bindings is recommended. Currently we provide support for Perl (Lorenz::API.pm) and JavaScript (Lorenz.js and LorenzJSONP.js).

Here is a complete list of REST endpoints. Unless otherwise noted (e.g., via a [PUT] notation), all request methods for these endpoints use GET.

REST Endpoint	Purpose
/bank/ /banks/	List all banks
/bank/<bank>	Show details about specific bank
/bank/<bank>/cpuutil	Show cpu utilization of each cluster for given bank on a daily basis
/chaos/	Show OS update information (downtime durations, comments)
/cluster/ /clusters/	List clusters on which jobs may be run
/cluster/<cluster>/bank/<bank>/cpuutil/daily	Show daily cpu utilization for given bank on given cluster
/cluster/<cluster>/batchdetails	Show details of the batch system on given cluster
/cluster/<cluster>/details	Show hardware and usage details of given cluster
/cluster/<cluster>/joblimits	Show job limits of given cluster
/cluster/<cluster>/topo	Return hardware topographical image of given cluster
/clusters/backfill	Show batch backfill information for each cluster

/clusters/details	Show hardware and usage details of all clusters
/command/<host>[POST]	Execute an arbitrary command on the given host. The command is run as the invoking user.
/email[POST]	Send an email as the invoking user.
/file/<host>/<path>	Retrieve the file located on host at given path
/group/<group>	Get information about the given group
/groups	List all groups
/host/ /hosts/	List all hosts in the center
/host/<host>	Show details about given host
/jsonp [GET]	Allows access to non-GET resources for JSONP applications. Use the 'route' and 'via' query arguments to describe the resource. E.g., /jsonp?via=delete&route=/file/aztec/var/tmp/junkfile
/lcalerts/	Retrieve current LC alerts
/lcalerts/<alertid>	Retrieve specific LC alert
/lcalerts/<alertid>[DELETE]	Delete given LC alert
/lcalerts/<alertid>[PUT]	Update an existing LC alert
/lcalerts/[PUT]	Create a new LC alert
/lcstaff/away	Access LC staff away calendar (must be in lcstaff group)
/lora/endpoints	List available endpoints
/news/	Return list of all news items
/news/<newsid>	Return specific news article
/news/ALL	Return all news articles
/noop	No-op
/parallelfs	List parallel filesystems
/portlet/getAllPortletConf	Get portlet configuration information for current user
/portlet/getCustomPortlets	Get custom portlets for current user
/queue/<host>	Show job queue on given host
/queue/<host>/<jobid>/STAT[GET]	Retrieve STAT-generated stack traces from given job
/queue/<host>/<jobid>/STAT[PUT]	Generate STAT stack trace for given job
/queue/<host>/<jobid>/steps	List job steps for given job
/queue/<host>/<jobid>[DELETE]	Cancel given job

/queue/<host>/<jobid>[GET]	Get details of given job
/queue/<host>/<jobid>[PUT]	Update properties of given queued or held job
/queue/<host>/reservation/<res>	Show details of given reservation on given host
/queue/<host>/reservations	List reservations on given host
/queue/<host>[POST]	Submit a batch job on given host
/scratchfs	List available scratch filesystems
/script/<host>/<script>[POST]	Execute a registered script on given host
/sqlog/<host>[POST]	Query historical slurm data using sqlog
/status/cluster /status/clusters	Show status of all clusters
/status/cluster/<cluster>	Show status of given cluster
/status/cluster/<cluster>/utilization/daily	Show cluster utilization on a daily level
/status/cluster/<cluster>/utilization/hourly	Show cluster utilization on an hourly level
/status/clusters/ME/utilization/daily	Show my daily cluster utilization
/status/clusters/ME/utilization/hourly	Show my hourly cluster utilization
/status/clusters/user/<user>	Show status of clusters to which given user has access
/status/clusters/utilization/daily	Show cluster utilization on a daily level
/status/clusters/utilization/daily2	Show cluster utilization on a daily level (alternate output format)
/status/clusters/utilization/hourly	Show cluster utilization on an hourly level
/status/clusters/utilization/hourly2	Show cluster utilization on an hourly level (alternate output format)
/status/filesystem	Show status of all filesystems
/status/host/<host>	Show status of given host
/status/license	Show status of all licenses
/status/license/<license>	Show status of given license
/status/loginNode	Show status of all login nodes
/status/machines	Show status of all hosts
/store	List available data stores for current user
/store/<store>[DELETE]	Delete user's given data store
/store/<store>[GET]	Retrieve user's given data store
/store/<store>[POST]	Update an existing data store for current user
/store/<store>[PUT]	Create new data store for current user
/support/defaultLinks	Return the default set of links
/support/getCredentialLifetime	Return the amount of time the current Kerberos credentials have before expiration

/support/lorenzAlert	Get the active Lorenz alert
/support/lorenzAlert[DELETE]	Remove the Lorenz alert
/support/lorenzAlert[PUT]	Create a Lorenz alert
/support/mylcToggle	Get the current mylc activation status
/support/mylcToggle[DELETE]	Activate a deactivated mylc site
/support/mylcToggle[PUT]	Deactivate an active mylc site
/support/network	Show the current network zone
/support/reportError[POST]	Report a Lorenz error to developers
/user/ /users/	List all users
/user/<user>	Show details of given user
/user/<user>/bank/<bank>	Show details of how given user can use given bank
/user/<user>/banks	Show user's banks
/user/<user>/cluster/<cluster>/batchdetails	Show batch details for given user on given cluster
/user/<user>/clusters	List clusters on which given user can submit a job
/user/<user>/cpuutil/daily	Show user's daily cpu utilization
/user/<user>/default/host	Show user's default host
/user/<user>/enclavestatus	Show user's Enclave status
/user/<user>/groups	Show groups to which given user belongs
/user/<user>/hosts	Show hosts on which given user has login accounts
/user/<user>/oun	Show given user's official username and LC username
/user/<user>/purgedFiles	Show files given user has had purged from Lustre in past 90 days
/user/<user>/queue	Show all of given user's running and queued jobs
/user/<user>/quotas	Show user's disk usage for various filesystems
/user/<user>/sshhosts	Show hosts to which given user can ssh to within the LC domain
/user/<user>/transferhosts	Show details related to file transfer hosts for given user
/user/ME/cache/<cache>[DELETE]	Delete a user's cached item
/user/ME/cache/<cache>[GET]	Retrieve a user's cached item
/user/ME/cache[DELETE]	Delete all of a user's cached items
/user/ME/cache[GET]	Get all of a user's cached items
/user/ME/give	Show current status as reported by 'give' command

/user/ME/give[POST]	Give a file to another LC user
/user/ME/givetake	Show current status as reported by 'give' and 'take' commands
/user/ME/take	Show current status as reported by 'take' command
/user/ME/take[POST]	Take a file which another LC user has given
/weather	Show current weather information

Notes on JSONP

Web applications hosted on remote servers can still take advantage of the Lorenz JavaScript API, but must deal with cross-site scripting (XSS) restrictions in the browser. To accommodate this use case, Lorenz provides support for JSONP, or "JSON with padding", a communication technique used in JavaScript. It provides a method to request data from a server in a different domain, something prohibited by typical web browsers because of the same origin policy. A limitation of this approach is that only the GET request method can be used when accessing the REST endpoints.

To address these issues, remote web app developers can use Lorenz-JSONP.js library, which has the same API as Lorenz.js, or can directly invoke the /jsonp endpoint.

If only accessing LORA endpoints that use the GET request method, the following instantiation can be used:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">

<html>
<head>
    <title>JSONP Test</title>
    <script src="https://lcwebdev.llnl.gov/lorenz/js/jquery/jquery-
1.8.2.min.js"></script>
    <script>
        $(document).ready(function(){
            var d = $.ajax({
                dataType: 'jsonp',
                url: 'https://lc.llnl.gov/lorenz/lora/lora.cgi/user/ME/groups'
            });

            d.done(function(){
               $('#output').html(JSON.stringify(arguments));
            });
        });
    </script>
</head>

<body>
    <div id="output"></div>
</body>
</html>
```

However, it is recommended to use the Lorenz-JSONP.js library instead, which maps PUT, POST, and DELETE methods behind the scenes to GET-equivalent functionality on the server.