SafeNet Network HSM 7.1

REST API User Guide

gemalto
security to be free

Document Information

Product Version	7.1
Document Part Number	<PartNumber>
Release Date	06 December 2017

Revision History

Revision	Date	Reason
A	06 December 2017	Initial release

Trademarks, Copyrights, and Third-party Software

Disclaimer

CONTENTS

PREFACE
About This Document

This document describes how to begin using the REST API and introduces a Python client development tool that promotes its automation. It contains the following chapters:

- "Overview" on page 8
- "Getting Started" on page 10
- "The REST API Sample Clients" on page 12
- "Using the REST API" on page 52

This preface also includes the following information about this document:

- "Customer Release Notes" below
- "Audience" below
- "Document Conventions" below
- "Support Contacts" on page 7

For information regarding the document status and revision history, see "Document Information" on page 2.

Customer Release Notes

The customer release notes (CRN) provide important information about this release that is not included in the customer documentation. It is strongly recommended that you read the CRN to fully understand the capabilities, limitations, and known issues for this release. You can view or download the latest version of the CRN for this release at the following location:

http://www.securedbysafenet.com/releasenotes/luna/crn_luna_rest_api_5-0-0.pdf.

Audience

This document is intended for personnel responsible for maintaining your organization's security infrastructure. This includes Luna HSM users and security officers, key manager administrators, and network administrators.

All products manufactured and distributed by Gemalto are designed to be installed, operated, and maintained by personnel who have the knowledge, training, and qualifications required to safely perform the tasks assigned to them. The information, processes, and procedures contained in this document are intended for use by trained and qualified personnel only.

It is assumed that the users of this document are proficient with security concepts.

Document Conventions

This document uses standard conventions for describing the user interface and for alerting you to important information.

Notes

Notes are used to alert you to important or helpful information. They use the following format:

Note: Take note. Contains important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss. They use the following format:

CAUTION: Exercise caution. Contains important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury. They use the following format:

WARNING! Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury.

Command Syntax and Typeface Conventions

Format	Convention
bold	The bold attribute is used to indicate the following: • Command-line commands and options (Type dir /p.) • Button names (Click Save As.) • Check box and radio button names (Select the Print Duplex check box.) • Dialog box titles (On the Protect Document dialog box, click Yes.) • Field names (User Name: Enter the name of the user.) • Menu names (On the File menu, click Save.) (Click Menu > Go To > Folders.) • User input (In the Date box, type April 1.)
italics	In type, the italic attribute is used for emphasis or to indicate a related document. (See the *Installation Guide* for more information.)
<variable>	In command descriptions, angle brackets represent variables. You must substitute a value for command line arguments that are enclosed in angle brackets.
[**optional**] [<optional>]	Represent optional **keywords** or <variables> in a command line description. Optionally enter the keyword or <variable> that is enclosed in square brackets, if it is necessary or desirable to complete the task.

Format	Convention

{**a**|**b**|**c**}
{<a>|<b>|<c>}
	Represent required alternate **keywords** or <variables> in a command line description. You must choose one command line argument enclosed within the braces. Choices are separated by vertical (OR) bars.

[**a**|**b**|**c**] [<a>|<b>|<c>]	Represent optional alternate keywords or variables in a command line description. Choose one command line argument enclosed within the braces, if desired. Choices are separated by vertical (OR) bars.

Support Contacts

If you encounter a problem while installing, registering or operating this product, please make sure that you have read the documentation. If you cannot resolve the issue, please contact your supplier or Gemalto support. Gemalto support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Gemalto and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Table 1: Technical support contacts

Contact method	Contact	
Address	Gemalto 4690 Millennium Drive Belcamp, Maryland 21017 USA	
Phone	United States	(800) 545-6608, (410) 931-7520
	Australia and New Zealand	+1 410-931-7520
	China	(86) 10 8851 9191
	France	0825 341000
	Germany	01803 7246269
	India	+1 410-931-7520
	United Kingdom	0870 7529200, +1 410-931-7520
Web	www.safenet-inc.com	
Support and Downloads	www.safenet-inc.com/support Provides access to the Gemalto Knowledge Base and quick downloads for various products.	
Technical Support Customer Portal	https://serviceportal.safenet-inc.com Existing customers with a Technical Support Customer Portal account can log in to manage incidents, get the latest software upgrades, and access the SafeNet Knowledge Base.	

1 Overview

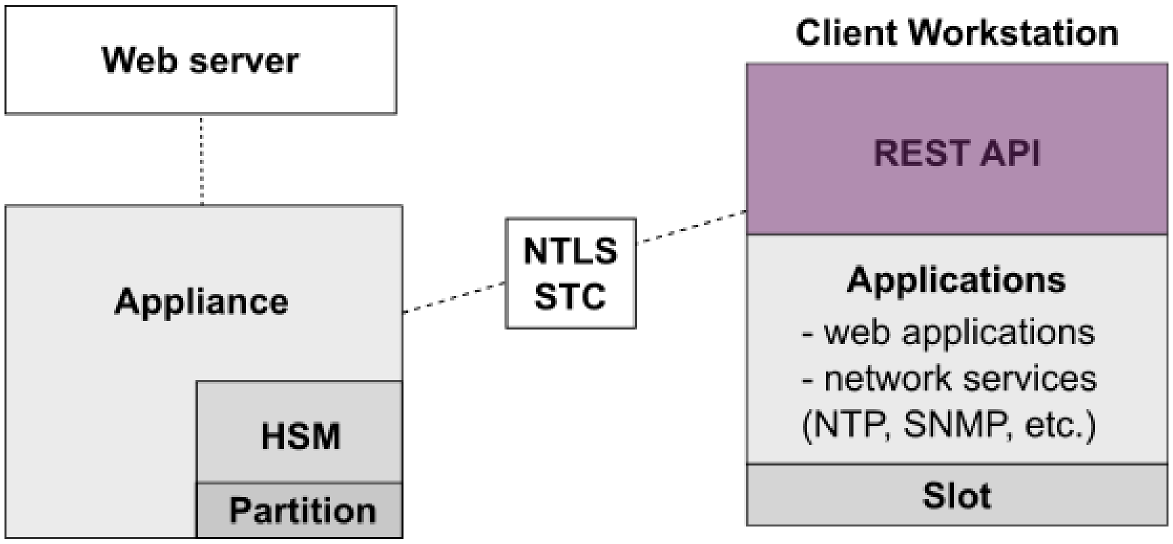
In addition to the long-standing Luna shell, administrators now have the ability to use a representational state transfer application programming interface — REST-ful API — to configure and query the appliance.

The REST API's advantage is its lightweight architecture. It is a simple mechanism that allows communication between SafeNet Network HSMs, servers, and applications. The REST API uses verb requests to retrieve, create, update, delete, and send data, as shown in "Connections facilitated by REST API" below.

The REST API client allows the user to perform these tasks all in one place, and serves as an organized demo application that helps users understand how the REST API works. It can be used as an alternate reference tool to the more comprehensive *REST API Command Reference* guide. It is particularly useful in program writing, as the REST architecture automates some of the manual work that had to be done in the Luna shell.

REST API Architecture

Figure 1: Connections facilitated by REST API



The function of the REST API is to facilitate communications between different appliances, servers, and applications. The REST API client allows users to do this all from one place and automates some of the manual work that had to be done when implementing the same operations in the Luna shell.

Secure Model

The REST API uses security protocols to ensure protection from malicious attacks while communicating sensitive information.

An open source cryptography library, OpenSSL, is used to implement SSL (Secure Sockets Layer) and TLS (Transport Security Layer) protocols to protect your data. SSL certificates are signed so that you can verify their source. Access to the REST API's operations is controlled through role-based authentication, to ensure that only authorized personnel can perform potentially destructive operations. Additionally, PKI authentication acts as a trusted party that validates the identity of individuals, computers, and other machines.

Data entry is limited. Early validation safeguards the server against the entry of rogue data. Likewise, the number of ciphers that can be entered is limited to prevent non-secure ciphers from intruding.

1 Getting Started

To use the REST API you must configure and enable the webserver on any SafeNet Luna Network HSMs you want to communicate with using the REST API, and install the SafeNet Luna client on any client workstation you want to use to communicate with a SafeNet Luna Network HSM using the REST API.

To configure and enable the webserver on a SafeNet Luna Network HSM

You must enable the web server on any SafeNet Luna Network HSMs you want to communicate with using the REST API.

1. Log in to the SafeNet Luna Network HSM as Admin.
2. Confirm that you have the correct REST API version installed. Luna release 7.1 requires REST API version 5.

   lunash:> **webserver show**
3. Set the REST API service to use a network port:

   lunash:> **webserver bind**
4. Enable the web server with the command:

   lunash:> **webserver enable**
5. Generate a certificate with the command:

   lunash:> **webserver certificate generate –keytype rsa**

   Note: It is recommended that you use the RSA algorithm for this cryptographic operation.
6. Configure the web server cipher suite, if necessary:

   lunash:> **webserver ciphers set -list** <cipher_list>
7. Restart the webserver service and test that the REST API is operational:

   lunash:> **service start webserver**

   Note: You can also restart the web server using the **-restart** option of any **webserver** command.
8. You may now begin using the API.

To configure your client workstation

1. It is assumed that you have the SafeNet HSM client installed and configured at your workstation. If not, please refer to the *SafeNet Network HSM Installation Guide*.

2. Use a service like NTLS or STC to connect to your HSM or partition in order to perform administrative and transformative operations.
3. Install the sample client, if desired. The client tool's purpose is to showcase the basic functions of the REST API in an organized format. It is an easy-to-follow development tool that acts as an interactive API call repository for those getting to know the API. You can use the client by referring to the sample code provided to you in the **client.zip** file. See "The REST API Sample Clients" on page 12 for more information.

2
The REST API Sample Clients

The REST API package includes a sample Python client and a sample Web client. These sample clients primarily serve as test tools to demonstrate interactions between the REST framework and communicating appliances. They are especially useful for new users .

The clients are organized to make running calls simple. They automate some of the more time-consuming tasks associated with manual queries, making REST interactions efficient. The client's clear and minimalistic layout allows new users to easily interact with the API, but keeps a custom query tab for those more familiar with its resources.

Calls are grouped under tabs representing different services a user may communicate with. User requests and responses are tracked in an output window at the bottom of the interface, making it easy to parse output and make decisions.

Note: Detailed structural information and schemata of the resources referenced in this document can be found in the *REST API Command Reference* documentation.

This section outlines the following:

- "Sample Workflow" below
- "Using the Python Client" on the next page
- "Using the Web Client" on page 36

Sample Workflow

A high-level workflow example in the REST API client is provided in this section.

Some basic operations you can perform with the REST API client include:

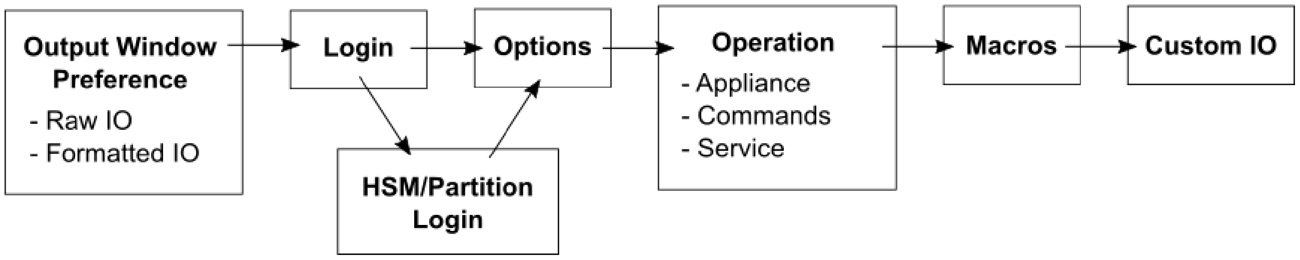
- Logging in to an HSM or partition directly or remotely,
- PED authentication and operation,
- Automation of services running with the REST API,
- Configuration and management of the appliance and other elements,
- Basic auditing and tracking, and
- Connection testing.

Each tab in the user interface contains corresponding resources and an output window at the bottom for tracking and parsing query results. For more complex or custom resource queries, a **Custom IO** tab is also included.

Note: Detailed structural information and schemata of the resources referenced in this document can be found in the *REST API Command Reference* documentation.

Getting Started

Figure 1: Workflow



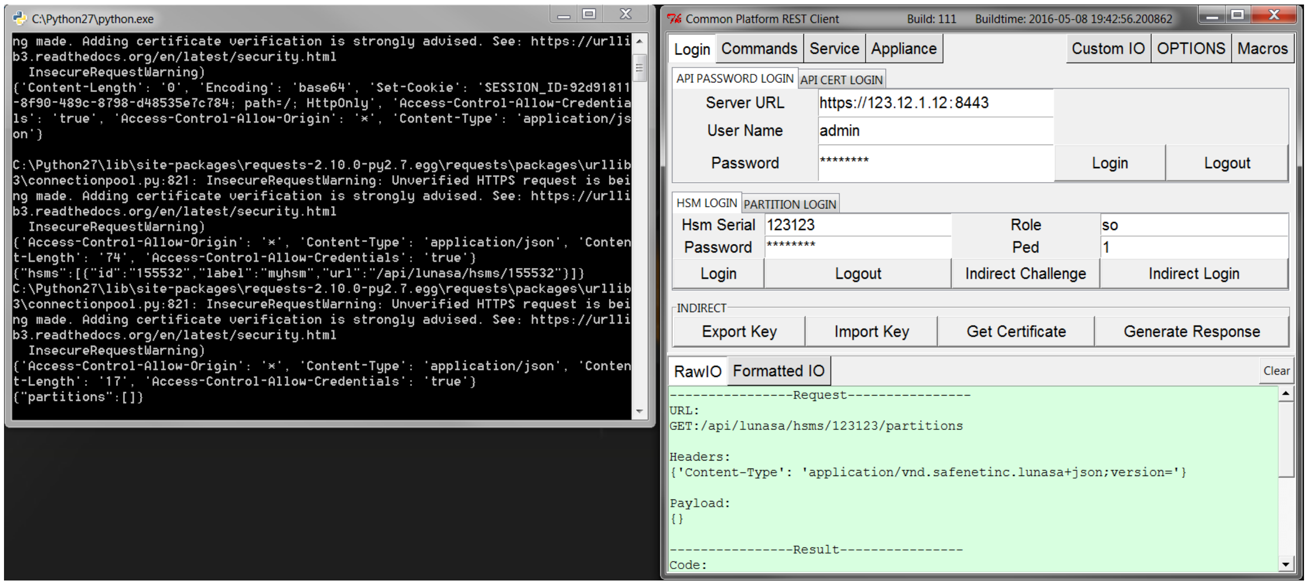
1. Choose **Raw IO** or **Formatted IO** from the output window at the bottom of the client to view the records of your requests and responses according to your preference.
   – **Raw IO** shows you your request input and output result. It turns green when a query is successful, and red when unsuccessful.
   – **Formatted IO** only shows the output result from your query, in an organized formatted view.
2. **Login** to your appliance to authenticate to the REST API. Depending on your permissions and what you wish to do with the REST API, you may also have to login to an HSM or partition.
3. Set your preferences for using the REST API in the **Options** tab.
4. Perform operations with your appliance, HSM (or partition), or services by using commands from their respective tabs.
   – Appliance tab: If you are not logged in to an HSM or partition, your commands are limited to configuration and logging from the **Appliance** tab.
   – **Commands** tab: Handles HSM or partition operations. If you are using PED-based authentication, connect to your PED. HSM or partition command control will be passed to the PED.
   – **Service** tab: If you are running any services alongside the REST API, commands to configure, start, stop, and set global preferences for them will be found here.
5. If you want to automate some parts of your operations with the REST API, use or create a macro instruction file in the **Macros** tab. Similarly, if you want to keep a log of steps you complete in a long process so that you can refer to it later, having a macro file is useful.
6. Once you become familiar with the fundamental resources available in the REST API client, use the **Custom IO** tab to customize your requests or input resources not available as client buttons.

Note: A complete list of resources you can query with the REST API can be found in the *REST API Command Reference* documentation.

Using the Python Client

The REST API Python client opens alongside a Python command window that tracks the usage of the client, as seen in "User Interface" on the next page.

Figure 1: User Interface



```
ng made. Adding certificate verification is strongly advised. See: https://urlli
b3.readthedocs.org/en/latest/security.html
  InsecureRequestWarning)
{'Content-Length': '0', 'Encoding': 'base64', 'Set-Cookie': 'SESSION_ID=92d91811
-8f90-489c-8798-d48535e7c784; path=/; HttpOnly', 'Access-Control-Allow-Credentia
ls': 'true', 'Access-Control-Allow-Origin': '*', 'Content-Type': 'application/js
on'}

C:\Python27\lib\site-packages\requests-2.10.0-py2.7.egg\requests\packages\urllib
3\connectionpool.py:821: InsecureRequestWarning: Unverified HTTPS request is bei
ng made. Adding certificate verification is strongly advised. See: https://urlli
b3.readthedocs.org/en/latest/security.html
  InsecureRequestWarning)
{'Access-Control-Allow-Origin': '*', 'Content-Type': 'application/json', 'Conten
t-Length': '74', 'Access-Control-Allow-Credentials': 'true'}
{"hsms":[{"id":"155532","label":"myhsm","url":"/api/lunasa/hsms/155532"}]}
C:\Python27\lib\site-packages\requests-2.10.0-py2.7.egg\requests\packages\urllib
3\connectionpool.py:821: InsecureRequestWarning: Unverified HTTPS request is bei
ng made. Adding certificate verification is strongly advised. See: https://urlli
b3.readthedocs.org/en/latest/security.html
  InsecureRequestWarning)
{'Access-Control-Allow-Origin': '*', 'Content-Type': 'application/json', 'Conten
t-Length': '17', 'Access-Control-Allow-Credentials': 'true'}
{"partitions":[]}
```

Server URL: https://123.12.1.12:8443
User Name: admin
Password: ********

Hsm Serial: 123123
Password: ********
Role: so
Ped: 1

```
----------------Request----------------
URL:
GET:/api/lunasa/hsms/123123/partitions

Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version='}

Payload:
{}

----------------Result----------------
Code:
```

Along the top of the client dialog box are tabs containing clusters of commands the REST API can perform. Underneath each tab are buttons and prompts for information related to the commands. Clicking on a button will run its corresponding resource and generate a response. Some commands will open dialog boxes, prompting you for more information.

Along the bottom of the client box is an output window that displays the code equivalent of your input requests and output results. It tracks all of your client actions, and can be cleared at any time by pressing the **Clear** button. Which tab you use to monitor your actions is up to you.

- **Raw IO** shows you your request (input) and result (output). The tab turns green when a query is successful, and red when unsuccessful.
- **Formatted IO** only shows the result (output) of your query, but does so in an organized view.

The REST API client is compatible with all versions of the API. By default, it uses the most recent version but it is possible to use a previous one without issues. You can change versions globally in the **Options** tab or per call in the **Custom IO** tab.

Comprehensive descriptions of each tab and its contents are contained in the following:

- "Login" below
- "Commands" on page 17
- "Service" on page 22
- "Appliance" on page 29
- "Custom IO" on page 32
- "Options" on page 33
- "Macros" on page 35

Login

To begin using the REST API, you must first login.

The client provides several different login options under the **Login** tab. The top two options, **API Password Login** and **API Cert Login**, login to the API on the appliance you are using. **API Password Login** uses password-based authentication, while **API Cert Login** uses a certificate.

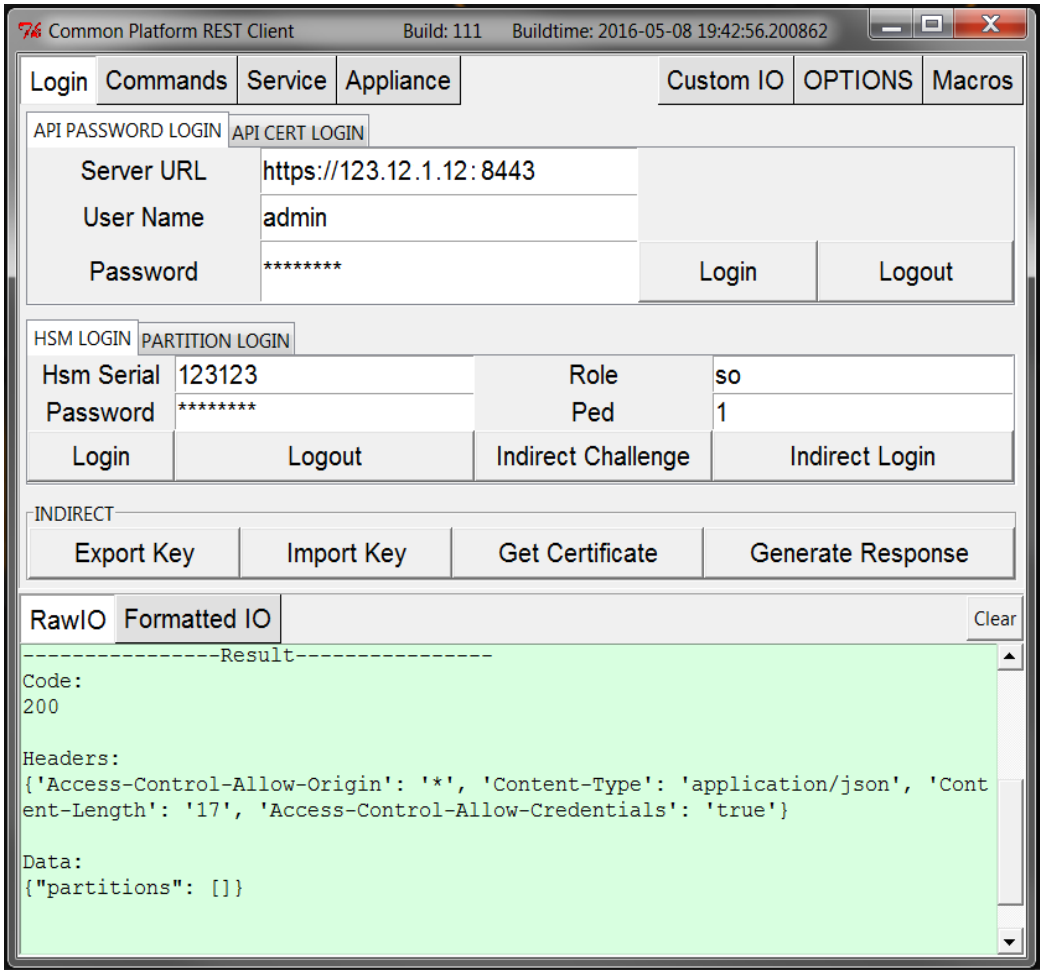
To login:

1. Fill in the **Server URL** box with your IP address.
2. Type your user name into the **User Name** box. The default user is **admin** (Administrator).
3. Type your password or load your certificate, depending on which option you are using.
4. Click **Login**.

If you are successful, the **Raw IO** tab at the bottom will turn green. If you are using the **Formatted IO** tab to track your outputs, you receive a success response. See "Login" on the previous page

If you are unsuccessful, the **Raw IO** tab will turn red and return an error. The **Formatted IO** tab will fail to populate with defined values, returning an error.

Figure 1: Login Tab showing successful login on Raw IO



Server URL: https://123.12.1.12:8443
User Name: admin
Password: ********

Hsm Serial: 123123
Password: ********
Role: so
Ped: 1

```
----------------Result----------------
Code:
200

Headers:
{'Access-Control-Allow-Origin': '*', 'Content-Type': 'application/json', 'Cont
ent-Length': '17', 'Access-Control-Allow-Credentials': 'true'}

Data:
{"partitions": []}
```

To perform any commands with the REST API, you must first login to the appliance.

The tabs underneath **API Password Login** and **API Cert Login** are for logging in to your HSM or partition. Login to your HSM or partition is required only if you are authorized to use those elements.

For detailed instructions for logging in to the appliance and/or your HSM or partition, see:

- "Appliance Login" below
- "HSM or Partition Login" below

Appliance Login

Appliance login is always required to use the REST API. However, appliance login alone restricts you to commands that manage your appliance.

API Password Login

If you are using password-based authentication with your appliance, use this method.

Input your IP address, username, and password. Click **Login**.

When you are finished using the REST API, or if you need to login to a different server or as a different user, click **Logout**.

API Cert Login

If you want to login using a certificate, use this method.

1. Input your IP address and username.
2. Generate a new certificate by clicking **Generate Cert**.
3. Once your certificate has been successfully generated, click **Load Cert** to specify the path to the certificate.
4. Click **Upload Cert** to establish and accept the certificate's association with the specified user (User Name).
5. Once an association is established, click **Login** to communicate this information to the server and secure the relationship.

When you are finished using the REST API, or if you need to login to a different server or as a different user, click **Logout**.

HSM or Partition Login

You must login to your HSM or partition if you want to make full use of the REST API.

HSM Login

There are two ways to login, depending on how you typically access your HSM.

- If you have direct access to your HSM and are using password-based authentication, input your HSM serial number and password. Click **Login**.
- If you have indirect access to your HSM via PED, input your user role and PED identifier number. Use 0 if you are using local PED; 1 or greater for remote PED. Click **Indirect Login**. Use the PED to complete your login.

When you are finished using the REST API, or if you need to login to a different server or as a different user, click **Logout**.

Partition Login

There are two ways to login, depending on how you typically access your partition.

- If you have direct access to your partition on an HSM and are using password-based authentication, input your partition serial number and password. Click **Login**.
- If you have indirect access to your partition via PED, input your user role and PED identifier number. Use 0 if you are using local PED; 1 or greater for remote PED. Click **Indirect Login**. Use the PED to complete your login.

When you are finished using the REST API, or if you need to login to a different server or as a different user, click **Logout**.

Indirect

If you want to indirectly authenticate to your HSM by way of another HSM, this is the method to use.

Note: This method requires two HSMs and is typically only used to perform an unattended login to a farm of HSMs.

Below, "Admin HSM" refers to the HSM you have direct access to; "Target HSM" refers to the target HSM that you are authenticating indirectly.

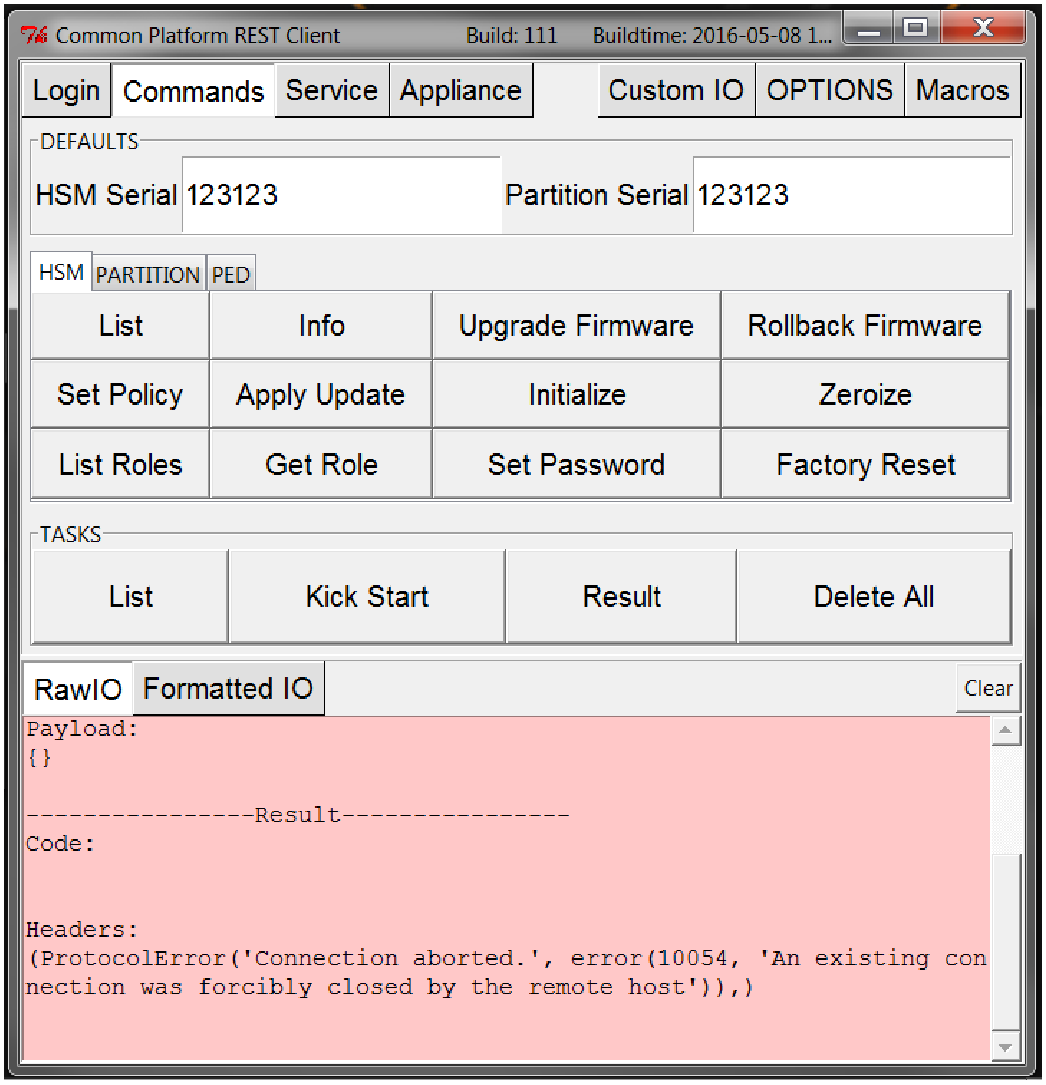
1. **Login** as co (crypto officer) to the partition on the Admin HSM that contains the indirect login key you wish to use.
2. Click **Export Key** to get the key from the Admin HSM.
3. **Login** as so (security officer) to the Target HSM.
4. Click **Import Key** to load the indirect login key onto the Target HSM. **Logout** of the Target HSM when finished.
5. Click **Get Certificate** to obtain the certificate needed for indirect login.
6. Click **Indirect Challenge** to get the indirect login challenge from the Admin HSM.
7. Click **Generate Response** to generate the indirect login response needed to communicate with the Target HSM.
8. Finally, click **Indirect Login** to indirectly login as so (security officer) to the Target HSM.

When you are finished using the REST API, or if you need to login to a different server or as a different user, click **Logout**.

Commands

The **Commands** tab contains most of the operations you will be performing with the REST API. The **Defaults** section running along the top contains the HSM and partition serial numbers you entered on the **Login** tab (see"Commands" above). If you wish to use a different HSM or partition, change these values.

Figure 1: Commands Tab showing an error on Raw IO



HSM Serial 123123 Partition Serial 123123

```
Payload:
{}

----------------Result----------------
Code:


Headers:
(ProtocolError('Connection aborted.', error(10054, 'An existing con
nection was forcibly closed by the remote host')),)
```

To perform a command:

1. Specify the HSM or partition you want to operate on by entering its serial number under **Defaults**.
2. Select the element you want to communicate with (**HSM**, **Partition**, or **PED**).
3. Click any command button to initiate its corresponding resource.

If the query is successful, the **Raw IO** tab at the bottom will turn green and show you a record of your request and its response. If you are using the **Formatted IO** tab to track your outputs, you only see the response values.

If you are unsuccessful, the **Raw IO** tab will turn red and return an error. The **Formatted IO** tab will fail to populate with defined values, returning an error.

Example

For example, to initialize your HSM:

1. Type the HSM serial number under **Defaults**.

2. Select the **HSM** tab.
3. Click **Initialize**.

The tables below list each command button on the **Commands** tab. Each command has a short description of what it does as well as its corresponding resource. The resources can be input manually in the **Custom IO** tab if you become very familiar with them.

Note: There are some calls that are not included as buttons in the client. They must be input manually in the **Custom IO** tab. A complete list of resources you can query with the REST API can be found in the *REST API Command Reference* documentation.

Commands are grouped below by applicable element. Calls to time-consuming resources are listed under **Tasks**.

- "HSM" below
- "Partition" on the next page
- "PED" on page 21
- "Tasks" on page 21

HSM

The table below defines commands you can send to your HSM and references their corresponding resources.

Command	Function	Resource
List	Lists all HSMs associated with appliance.	GET /api/lunasa/hsms
Info	Gets information associated with a specific HSM.	GET /api/lunasa/hsms/{hsmid}
Upgrade Firmware	Updates HSM firmware to the most recent version.	POST /api/lunasa/hsms/{hsmid}/firmware/actions/upgrade
Rollback Firmware	Downgrades HSM firmware to a previously installed version.	POST /api/lunasa/hsms/{hsmid}/firmware/actions/rollback
Set Policy	Sets a specific HSM policy.	PUT /api/lunasa/hsms/{hsmid}/policies/{policyid}
Apply Update	Applies a specific HSM update.	POST /api/lunasa/hsms/{hsmid}/updates/{updateid}
Initialize	Initializes the HSM.	PUT /api/lunasa/hsms/{hsmid}/
Zeroize	Removes all partitions and keys from the HSM. Does not reset HSM policies, erase RPV, or delete Auditor role.	POST /api/lunasa/hsms/{hsmid}/actions/zeroize
List Roles	Lists all roles associated with the HSM.	GET /api/lunasa/hsms/{hsmid}/roles
Get Role	Gets information associated with a specific HSM role.	GET /api/lunasa/hsms/{hsmid}/roles/{roleid}

Command	Function	Resource
Set Password	Opens a dialog box in which you can set a new HSM role password by following these steps: 1. Complete the form with your new password and old password. 2. Optionally change secret and/or challenge secret associated with a particular HSM or role by changing **false** to **true** and specifying HSM serial number and role.	PATCH /api/lunasa/hsms/{hsmid}/roles/{roleid}
Factory Reset	Sets the HSM back to its factory default settings. ⚠ **CAUTION:** Deletes the HSM SO, all users, and all objects.	POST /api/lunasa/hsms/{hsmid}/actions/factoryReset

Partition

The table below defines commands you can send to your partition and references their corresponding resources.

Command	Function	Resource
List	Lists all partitions associated with the HSM.	GET /api/lunasa/hsms/{hsmid}/partitions
Info	Gets information associated with a specific partition.	GET /api/lunasa/hsms/{hsmid}/partitions/{partitionid}
Create	Creates a partition.	POST /api/lunasa/hsms/{hsmid}/partitions
Delete	Removes a specific partition from the HSM.	DELETE /api/lunasa/hsms/{hsmid}/partitions/{partitionid}
Set Policy	Sets a specific partition policy.	PUT /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/policies/{policyid}
Delete All	Removes all partitions from the HSM.	DELETE /api/lunasa/hsms/{hsmid}/partitions
Initialize	Initializes the partition. (Applicable only to PPSO partitions.)	PUT /api/lunasa/hsms/{hsmid}/partitions/{partitionid}
Initialize Role	Initializes a specified partition role.	PUT /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/roles/{roleid}
List Roles	Lists all roles associated with the partition.	GET /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/roles

Command	Function	Resource
Get Role	Gets information associated with a specific partition role.	GET /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/roles/{roleid}
Set Password	Opens a dialog box in which you can set a new partition role password by following these steps: 1. Complete the form with your new password and old password. 2. Optionally change secret and/or challenge secret associated with a particular HSM, partition, and/or role by changing **false** to **true** and specifying HSM serial number, partition serial number, and role.	PATCH /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/roles/{roleid}
Create Challenge	Opens a dialog box in which you can create a new challenge for the partition by following these steps: 1. Complete the form with your HSM serial number, partition serial number, and role. 2. Optionally change the default value from **true** to **false** to randomize the challenge value. **Note:** Applies to PED-based partitions	POST /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/roles/{roleid}/actions/createChallenge

PED

The table below defines commands you can use with your PED and references their corresponding resources.

Command	Function	Resource
Connect	Connects to a Remote PED.	POST /api/lunasa/hsms/{hsmid}/peds/{pedid}/actions/connect
Disconnect	Disconnects the currently active Remote PED.	POST /api/lunasa/hsms/{hsmid}/peds/{pedid}/actions/disconnect
Vector Init	1. Initializes a Remote PED Vector (RPV). 2. Creates a new Remote PED Key (RPK). 3. Imprints the RPV onto the HSM and the RPK.	POST /api/lunasa/hsms/{hsmid}/peds/{pedid}/actions/vectorInitialize
Vector Erase	Erases the Remote PED vector (RPV) from the current HSM.	POST /api/lunasa/hsms/{hsmid}/peds/{pedid}/actions/vectorErase

Tasks

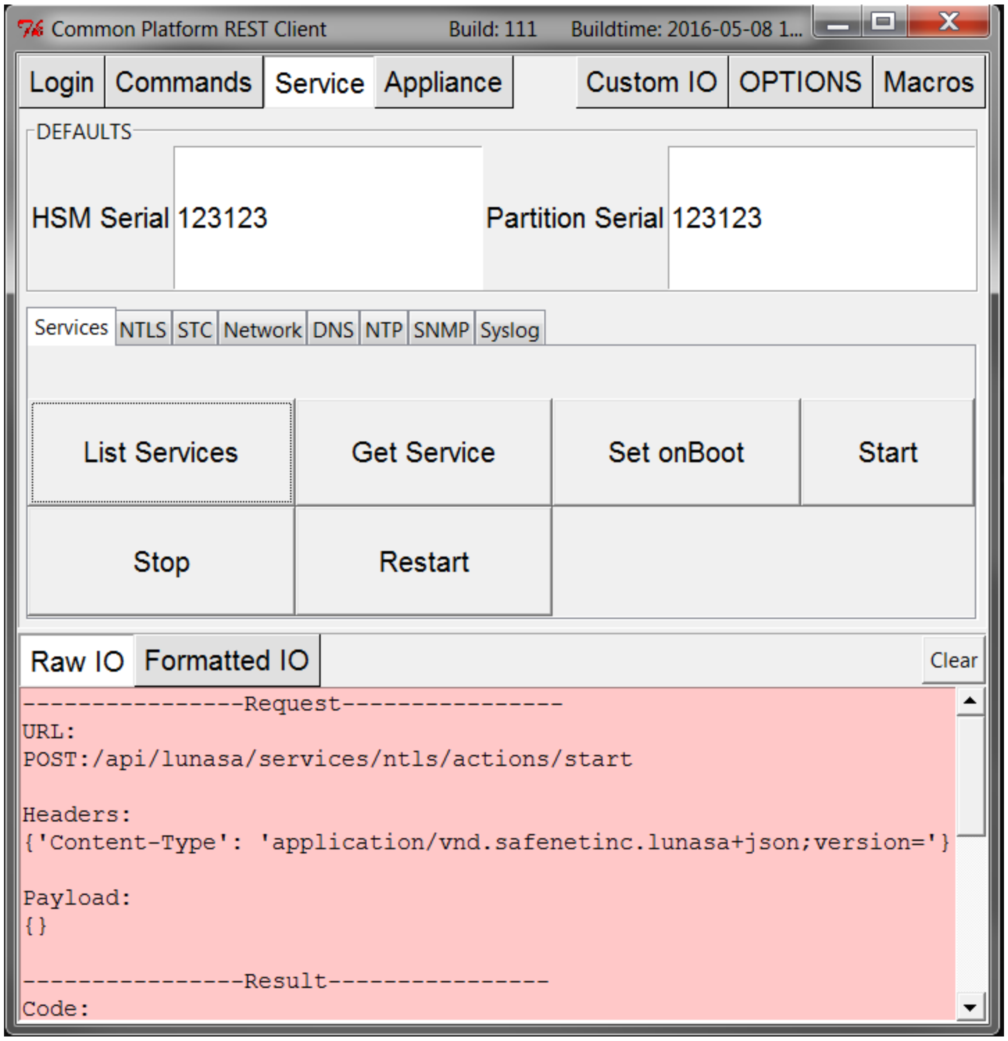
Tasks enable monitoring and administration of REST API resources that may require significant time to complete, such as updating HSM firmware. Rather than block and wait for these actions to complete, the REST API creates tasks to run the resource automatically in the background. Tasked resources return a response immediately and notify you of the status of the action: waiting, running, failed, etc. Because of their time-consuming nature, tasks are grouped separately. The table below defines each command and references its corresponding resource.

Command	Function	Resource
List	Lists all available tasked resources.	GET /tasks
Kickstart	Starts a waiting task.	POST /tasks/{taskid}/actions/start
Result	Gets the result and deletes the task.	GET /tasks/{taskid}/response
Delete All	Deletes all tasks.	DELETE /tasks

Service

Services are applications that work with REST API to communicate and manipulate information for external elements, like a server connected through a network. The **Service** tab contains operations you may perform with the services you are using. The **Defaults** section running along the top contains the HSM and partition serial numbers you entered on the **Login** tab (see "Service" above). If you wish to use a different HSM or partition, change these values.

Figure 1: Service Tab showing an error on Raw IO



HSM Serial 123123
Partition Serial 123123

```
----------------Request----------------
URL:
POST:/api/lunasa/services/ntls/actions/start

Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version='}

Payload:
{}

----------------Result----------------
Code:
```

To perform a command:

1. Specify the HSM or partition you want to operate on by entering its serial number under **Defaults**.
2. Select the service you want to communicate with (**NTLS**, **STC**, etc.), or click the **Services** tab to use more general commands and settings.
3. Click on a command button to initiate its corresponding resource.

If the query is successful, the **Raw IO** tab at the bottom will turn green and show you a record of your request and its response. If you are using the **Formatted IO** tab to track your outputs, you only see the response values.

If you are unsuccessful, the **Raw IO** tab will turn red and return an error. The **Formatted IO** tab will fail to populate with defined values, returning an error.

Example

For example, to stop a service on your partition:

1. Type in the HSM and partition serial numbers under **Defaults**.
2. Select the **Services** tab.

3. Click **Stop**. A dialog box will appear.
4. Enter the service you wish to stop into the dialog box prompt and click **Ok**.

The tables below list each command button you see on the **Service** tab in the client. Each command has a short description of what it does as well as its corresponding resource. The resources can be input manually in the **Custom IO** tab if you become very familiar with them.

Note: There are some calls that are not included as buttons in the client. They must be input manually in the **Custom IO** tab. A complete list of resources you can query with the REST API can be found in the *REST API Command Reference* documentation.

Commands are grouped by their corresponding service:

- "Services" below
- "NTLS" on the next page
- "STC" on the next page
- "Network" on page 26
- "DNS" on page 26
- "NTP" on page 27
- "SNMP" on page 27
- "Syslog" on page 28

Services

The **Services** tab sets general preferences for any or all services with the REST API.

The table below defines each **Services** command and references its corresponding resource.

Command	Function	Resource
List Services	Lists all services associated with the appliance.	GET /api/lunasa/services
Get Service	Gets information on a specified service.	GET /api/lunasa/services/{serviceid}
Set onBoot	Sets a specified service to be running on startup.	PUT /api/lunasa/services/{serviceid}
Start	Starts the named service.	POST /api/lunasa/services/{serviceid}/actions/start
Stop	Stops the named service.	POST /api/lunasa/services/{serviceid}/actions/stop
Restart	Restarts the named service.	POST /api/lunasa/services/{serviceid}/actions/restart

NTLS

Network Trust Link Service (NTLS), guarantees a secure connection when transferring data over a network. It encrypts your data and uses two-way digital certificate authentication to protect sensitive information so that you can ensure the security of your proprietary information.

The table below defines each **NTLS** command and references its corresponding resource.

Command	Function	Resource
List Clients	Lists all NTLS clients registered with the appliance.	GET /api/lunasa/ntls/clients
Register Client	Registers a client with the appliance.	POST /api/lunasa/ntls/clients
Assign Partition	Registers a client with a partition on the HSM.	POST /api/lunasa/ntls/clients/{clientid}/partitions
Delete Client	Deletes the specified client from the appliance.	DELETE /api/lunasa/ntls/clients/{clientid}
Get Server Cert	Gets the server-side certificate used by NTLS to establish connections with clients.	GET /api/lunasa/ntls/certificate
List Partitions	Lists all partitions registered to a specified client.	GET /api/lunasa/ntls/clients/{clientid}/partitions

STC

Secure Trusted Channel (STC), guarantees privacy and security in user-to-HSM communications. STC uses encryption, message authentication codes, and bi-directional endpoint authentication to ensure that only those authorized to use the connection can do so, and that your messages remain protected.

The table below defines each **STC** command and references its corresponding resource.

Command	Function	Resource
Register Stc Client	Registers a client identity for secure trusted communication with a specified partition.	POST /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/stc/clients
List Stc Client	Lists all client identities associated with the secure trusted channel for the specified partition.	GET /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/stc/clients
Export Partition	Exports the specified partition's public key to a file.	GET /api/lunasa/hsms/

Command	Function	Resource
		{hsmid}/partitions/ {partitionid}/stc

Network

The **Network** tab manages all your network devices and your connection to them.

The table below defines each **Network** command and references its corresponding resource.

Command	Function	Resource
Network Info	Gets the network information associated with the appliance.	GET /api/lunasa/network
Set Network Info	Sets all base network configurations associated with the appliance.	PUT /api/lunasa/network
List Devices	Lists all network devices.	GET /api/lunasa/network/devices
Get Device	Gets information on the specified network device.	GET /api/lunasa/network/devices/ {deviceid}
Get Device IP4	Gets IP4 information on the specified network device.	GET /api/lunasa/network/devices/ {deviceid}/ip4
Change Device	Changes the network device in use.	PATCH /api/lunasa/network/
Ping	Tests network connectivity to host.	POST /api/lunasa/network/actions/ping

DNS

You manage your DNS, or Domain Name Server, in the **DNS** tab.

The table below defines each **DNS** command and references its corresponding resource.

Command	Function	Resource
List nameSRVs	Lists currently registered name servers.	GET /api/lunasa/network/dns/nameServers
Create nameSRV	Creates a new name server.	POST /api/lunasa/network/dns/nameServers
Get nameSRV	Gets information on a specified name server.	GET /api/lunasa/network/dns/nameServers/ {nameServerid}

Command	Function	Resource
Delete nameSRV	Deletes a name server entry.	DELETE /api/lunasa/network/dns/nameServers/{nameServerid}
List searchDOMs	Lists currently registered search domains.	GET /api/lunasa/network/dns/searchDomains
Create searchDOM	Creates a new search domain.	POST /api/lunasa/network/dns/searchDomains
Get searchDOM	Gets information on a specified search domain.	GET /api/lunasa/network/dns/searchDomains/{searchDomainid}
Delete searchDOM	Deletes a search domain entry.	DELETE /api/lunasa/network/dns/searchDomains/{searchDomainid}

NTP

Network Time Protocol (NTP), provides connections to highly accurate time data servers so that your appliance can be synchronized. All devices can undergo gradual time drifts, and correcting these drifts with NTP is essential for applications to run smoothly.

The table below defines each **NTP** command and references its corresponding resource.

Command	Function	Resource
Get NTP	Gets NTP configuration information.	GET /api/lunasa/ntp
List Servers	Lists current server resources.	GET /api/lunasa/ntp/servers
Add Server	Adds an NTP server.	POST /api/lunasa/ntp/servers
Delete Server	Deletes a server entry.	DELETE /api/lunasa/ntp/servers/{serverid}
Server Info	Gets information on a specified server.	GET /api/lunasa/ntp/servers/{serverid}
Get Status	Returns information on ntp time, max error, estimated error, and offset.	GET /api/lunasa/ntp/status
Synchronize	Synchronizes date and time with NTP.	POST /api/lunasa/ntp/actions/synchronize

SNMP

Simple Network Management Protocol (SNMP), monitors a local HSM for changes in certain conditions that may cause problems. Traps can be put in place to respond to these condition changes and notify the appropriate personnel of errors in functioning.

The table below defines each **SNMP** command and references its corresponding resource.

Command	Function	Resource
Get SNMP	Gets SNMP configuration information.	GET /api/lunasa/snmp
Trap Info	Gets SNMP trap configuration information.	GET /api/lunasa/snmp/trap
Configure Trap	Configures SNMP trap parameters.	PUT /api/lunasa/snmp/trap
Delete Trap	Clears SNMP configuration.	DELETE /api/lunasa/snmp/trap
List Users	Lists SNMP users.	GET /api/lunasa/snmp/users
Create User	Creates an SNMP user.	POST /api/lunasa/snmp/users
User Info	Gets configuration information of a specified user.	GET /api/lunasa/snmp/users/{userid}
Delete User	Deletes a user.	DELETE /api/lunasa/snmp/users/{userid}
List Notifications	Lists SNMP notifications for a specified user.	GET /api/lunasa/snmp/users/{userid}/notifications
Create Notification	Creates an SNMP user notification.	POST /api/lunasa/snmp/users/{userid}/notifications
Notification Info	Gets configuration information for a specified notification.	GET /api/lunasa/snmp/users/{userid}/notifications/{notificationid}
Delete Notification	Deletes a specified notification.	DELETE /api/lunasa/snmp/users/{userid}/notifications/{notificationid}

Syslog

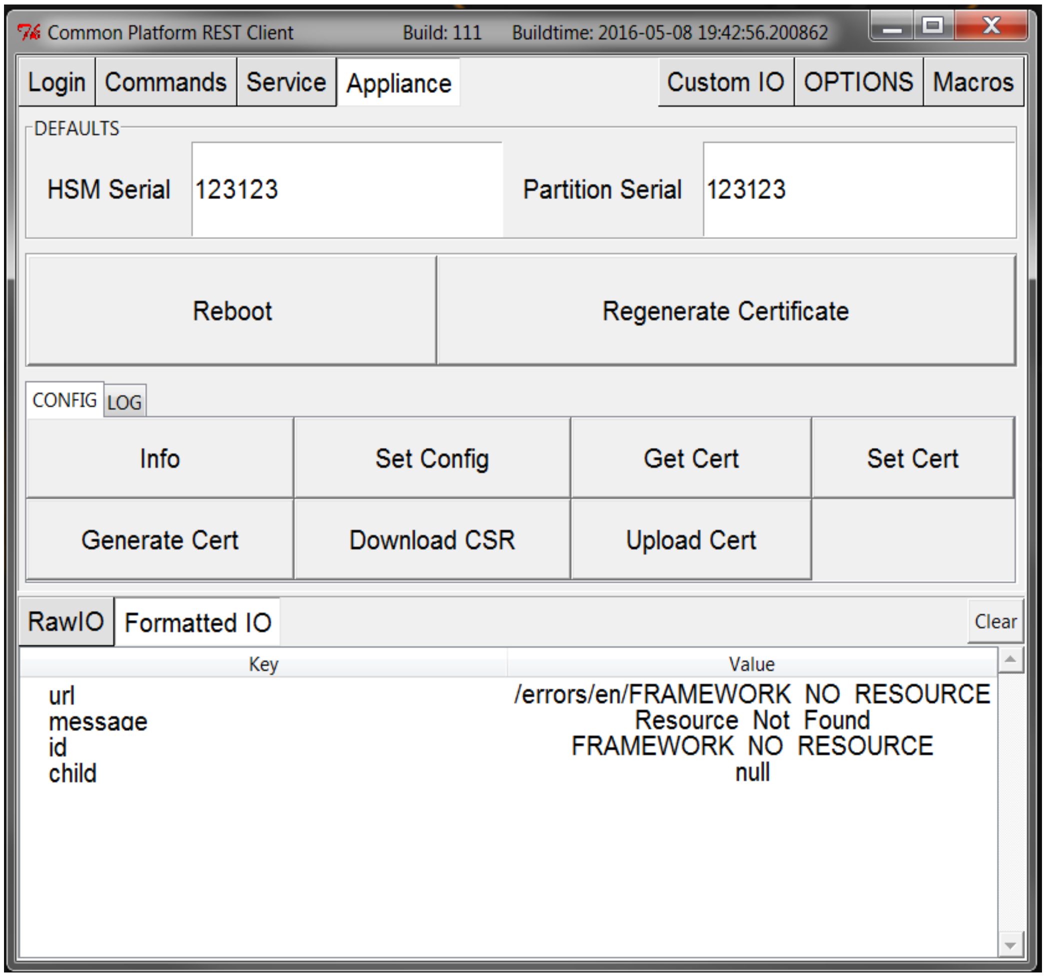
The table below defines each **Syslog** command and references its corresponding resource.

Command	Function	Resource
List Backups	Lists stored syslog backups.	GET /api/lunasa/syslog/backups
Create Backup	Creates a syslog backup.	POST /api/lunasa/syslog/backups
Get Backup	Retrieves a syslog backup and deletes it after.	GET /api/lunasa/syslog/backups/{backupid}
Delete Backup	Deletes a specified syslog backup.	DELETE /api/lunasa/syslog/backups/{backupid}
List remoteHosts	Lists configured remote hosts.	GET /api/lunasa/syslog/remoteHosts
Create remoteHost	Creates a remote host entry.	POST /api/lunasa/syslog/remoteHosts
remoteHost Info	Gets information on a specified remote host.	GET /api/lunasa/syslog/remoteHosts/{remoteHostid}
Delete remoteHost	Deletes specified remote host entries.	DELETE /api/lunasa/syslog/remoteHosts/{remoteHostid}

Appliance

The **Appliance** tab contains appliance-level administration commands. If you can only login to your appliance, and not to an HSM or partition, these are the available operations. If you are able to access an HSM or partition, the HSM and partition serial numbers you entered on the **Login** tab appear in the **Defaults** section running along the top (see "Appliance Tab showing an error on Formatted IO" on the next page). If you wish to use a different HSM or partition, change these values.

Figure 1: Appliance Tab showing an error on Formatted IO



Key	Value
url	/errors/en/FRAMEWORK NO RESOURCE
message	Resource Not Found
id	FRAMEWORK NO RESOURCE
child	null

To perform a command:

1. Select a basic action you want to perform, or a tab to view other tasks you can launch.
2. Click on a command button to initiate its corresponding resource.

If the query is successful, the **Raw IO** tab at the bottom will turn green and show you a record of your request and its response. If you are using the **Formatted IO** tab to track your outputs, you only see the response values.

If you are unsuccessful, the **Raw IO** tab will turn red and return an error. The **Formatted IO** tab will fail to populate with defined values, returning an error.

Note: Login to an HSM or partition is not necessary to perform operations on your appliance.

Example

For example, if you want to download a Certificate Signing Request (CSR):

1. Select the **Config** tab.
2. Click **Download CSR**.

The tables below list each command button you see on the **Appliance** tab in the client. Each command has a short description of what it does as well as its corresponding resource. The resources can be input manually in the **Custom IO** tab if you become very familiar with them.

Note: There are some calls that are not included as buttons in the client. They must be input manually in the **Custom IO** tab. A complete list of resources you can query with the REST API can be found in the *REST API Command Reference* documentation.

Commands are grouped by type:

- "Basic" below
- "Config" below
- "Log" on the next page

Basic

Reboot	Performs a warm restart (reboot) of the appliance, shutting down all running processes in a controlled manner.
Regenerate Certificate	Deletes and replaces your certificate with a newly generated one.

Config

The table below defines each **Config** command and references its corresponding resource.

Command	Function	Resource
Info	Gets configuration information of the web server providing the REST API.	GET /api/lunasa/webServer
Set Config	Sets the configuration of the web server providing the REST API. Complete the form with the network devices you want to use, threads information, port number, and your list of ciphers.	PATCH /api/lunasa/webServer
Get Cert	Gets attributes of the certificate.	GET /api/lunasa/webServer/certificate
Set Cert	Sets the certificate.	PATCH /api/lunasa/webServer/certificate
Generate Cert	Generates a new certificate.	POST /api/lunasa/webServer/certificate/actions/regenerate
Download CSR	Downloads a Certificate Signing Request.	GET /api/lunasa/webServer/csr
Upload Cert	Replaces the current certificate with a given file.	PUT /api/lunasa/webServer/certificate

Log

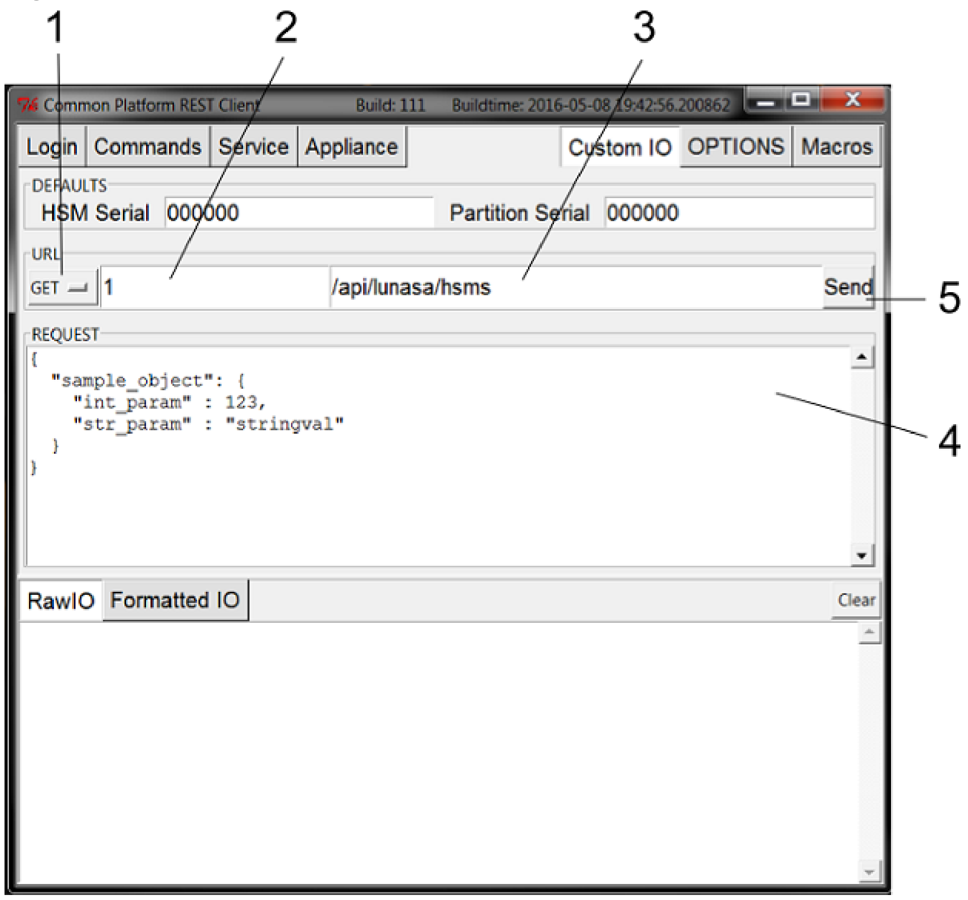
The table below defines each **Log** command and references its corresponding resource.

Command	Function	Resource
Download	Downloads the logs accumulated on the appliance.	GET /api/lunasa/logs
Send	Creates a log record on the appliance and sends it to lunalogs.	POST /api/lunasa/logs

Custom IO

The **Custom IO** tab allows you to input requests manually. Instead of clicking a button to query a resource, you can specify a resource by its verb form and request a custom payload (see "Custom IO Tab" below). This tab is useful if you are familiar with the commands and do not want to switch through various tabs in the client's user interface, or if you want to query a resource not included as a button in the client.

Figure 1: Custom IO Tab



```
{
  "sample_object": {
    "int_param" : 123,
    "str_param" : "stringval"
  }
}
```

1. This tab lists verbs you can choose from.
2. The number in this box corresponds to the REST API version you want to run. If left blank, the latest version is used by default.
3. The address of your resource goes in this box.
4. The target payload is specified in the code within this box.

5. After filling in items 1-4, click **Send** to make a formal request.

The **Defaults** section running along the top contains the HSM and partition serial numbers you entered on the **Login** tab. If you wish to use a different HSM or partition, change these values.

Example

To change a specific HSM policy:

1. Specify the HSM you want to operate on by entering its serial number under **Defaults**.
2. From the drop-down verb list, select **PATCH**.
3. Leave the version box blank to use the latest REST API version.
4. Input your resource address, **/api/lunasa/hsms/{hsmid}/policies/{policyid}**, where {hsmid} is your HSM serial number and {policyid} is your policy number.
5. Edit your request using the schema in the **Request** box to obtain your desired output.
6. Click **Send**.

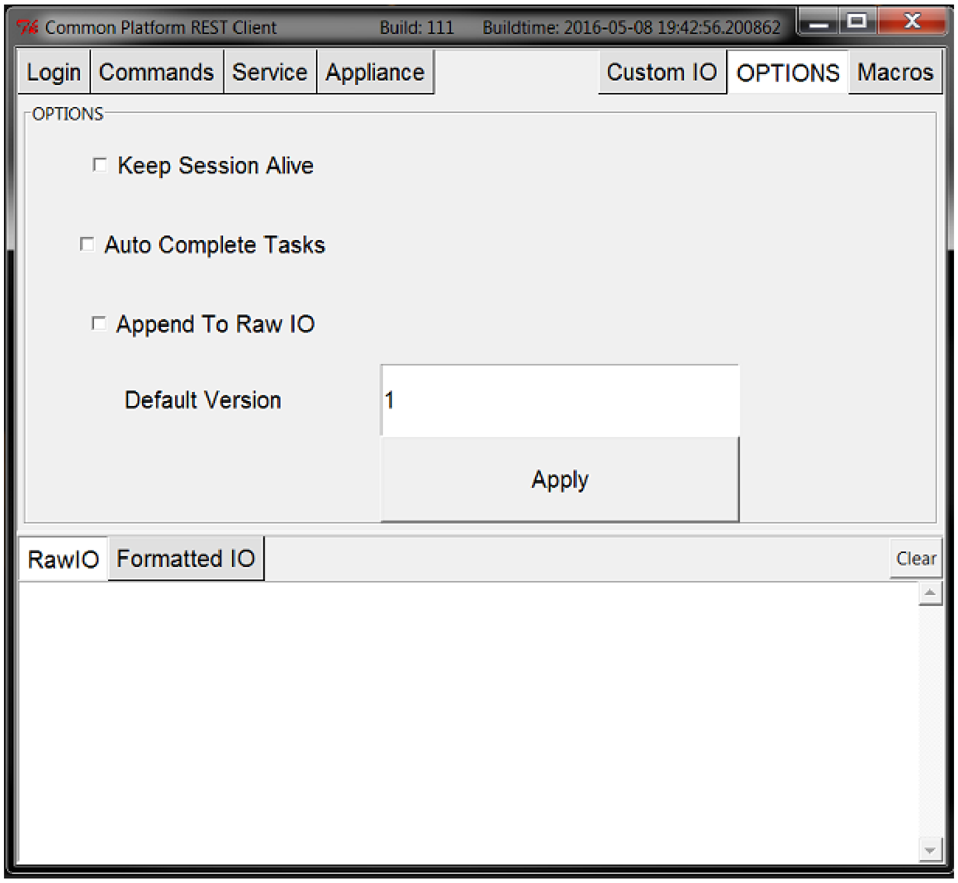
If the query is successful, the **Raw IO** tab at the bottom will turn green and show you a record of your request and its response. If you are using the **Formatted IO** tab to track your outputs, you only see the response values.

If you are unsuccessful, the **Raw IO** tab will turn red and return an error. The **Formatted IO** tab will fail to populate with defined values, returning an error.

Options

The **Options** tab contains selections you can invoke or remove, depending on your REST API use preferences (see "Options" above).

Figure 1: Options Tab



To change an option:

1. Select options you want to invoke; deselect those you do not want. You can use any combination of selections, including all or none.
2. Optionally set the default version of your REST API to one you specify.

Note: A default version set to 0 or -1 uses the most recent version of the REST API.

3. Click **Apply**.

If the query is successful, the **Raw IO** tab at the bottom will turn green and show you a record of your request and its response. If you are using the **Formatted IO** tab to track your outputs, you only see the response values.

If you are unsuccessful, the **Raw IO** tab will turn red and return an error. The **Formatted IO** tab will fail to populate with defined values, returning an error.

Options

Keep Session Alive	Prevents Login timeout by automatically logging into the REST API every 10 minutes.
Auto Complete Tasks	Completes Tasks automatically so that there is no need to manually monitor their status.

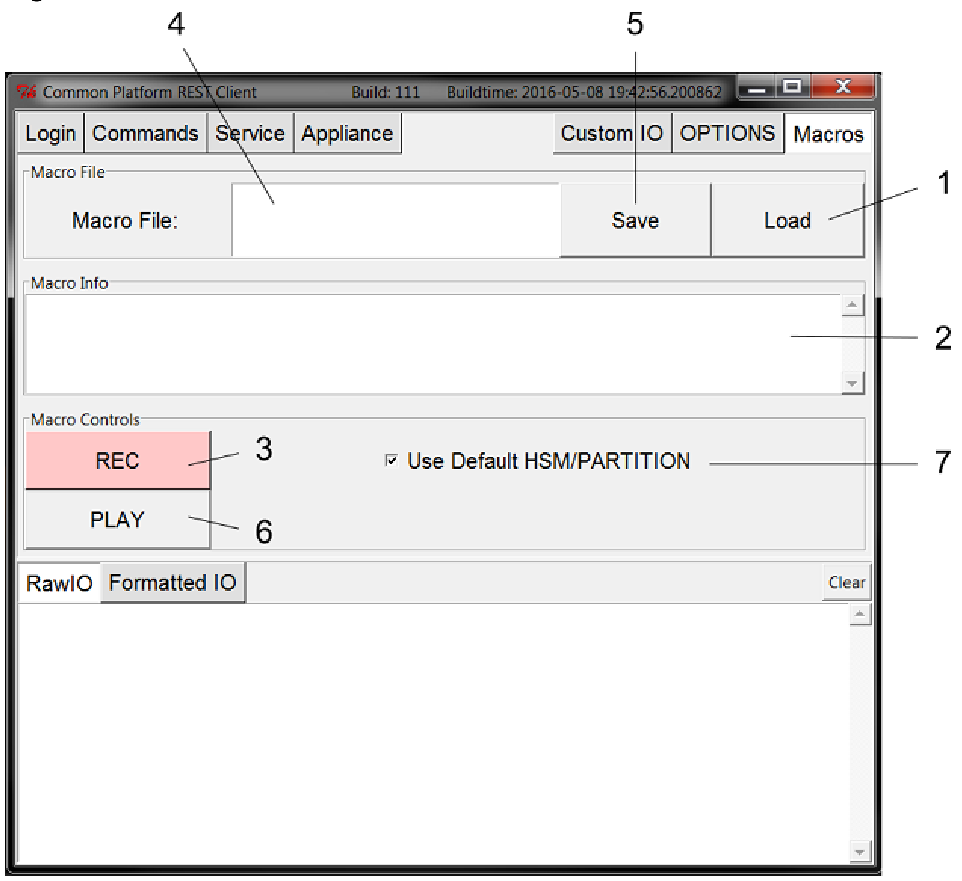
Append to Raw IO	Adds new requests to the **Raw IO** text without erasing previous data.
Default Version	Sets the default version of your REST API to one that you specify.

After enabling or disabling any of these features, you must click **Apply** for your changes to take effect.

Macros

The **Macros** tab allows you to automate your REST API use by using a macro instruction file (see"Macros" above). This tab is useful if you want to build or run a list of commands in a specific order, or multiple times as a test. You can also use Macros to ensure that a long list of commands is executed without manual input errors, saving time.

Figure 1: Macros Tab



1. **Load** specifies the path to your pre-existing macro file.
2. The commands specified in your macro file appear in the **Macro Info** box.
3. To create a new macro file, click **REC** to record your actions with the REST API. You can keep track of what you have done in the **Macro Info** box. Click **STOP REC** when finished.
4. This box allows you to name your newly-created macro file or rename a pre-existing file you loaded and may have edited.
5. To save a new or edited macro file, click **Save**. Specify your save path.
6. To execute the commands specified in your macro file, click **PLAY**.

7. **Use Default HSM/PARTITION** plays your macro file instructions on the default HSM or partition. Deselect it to play your macro file on a different HSM.

Example

To create a new macro file:

1. Click **REC** to begin recording your actions.
2. Click on a command button in any tab to initiate its corresponding resource. Continue inputting commands until you are satisfied with your output.
3. Go to the **Macros** tab. Click **STOP REC**.
4. You can view your input history in the **Macro Info** box to ensure you have the order of commands you want. You can also **PLAY** your newly-created file to check that it works properly.
5. Name your macro file in the box at the top.
6. Click **Save** to specify a path to where you want to save your macro file.

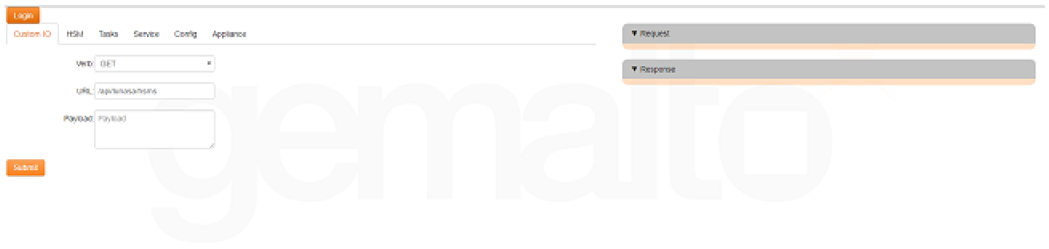
If you are successful, the **Raw IO** tab at the bottom will turn green and show you a record of your request and its response. If you are using the **Formatted IO** tab to track your outputs, you only see the response values.

If you are unsuccessful, the **Raw IO** tab will turn red and return an error. The **Formatted IO** tab will fail to populate with defined values, returning an error.

Using the Web Client

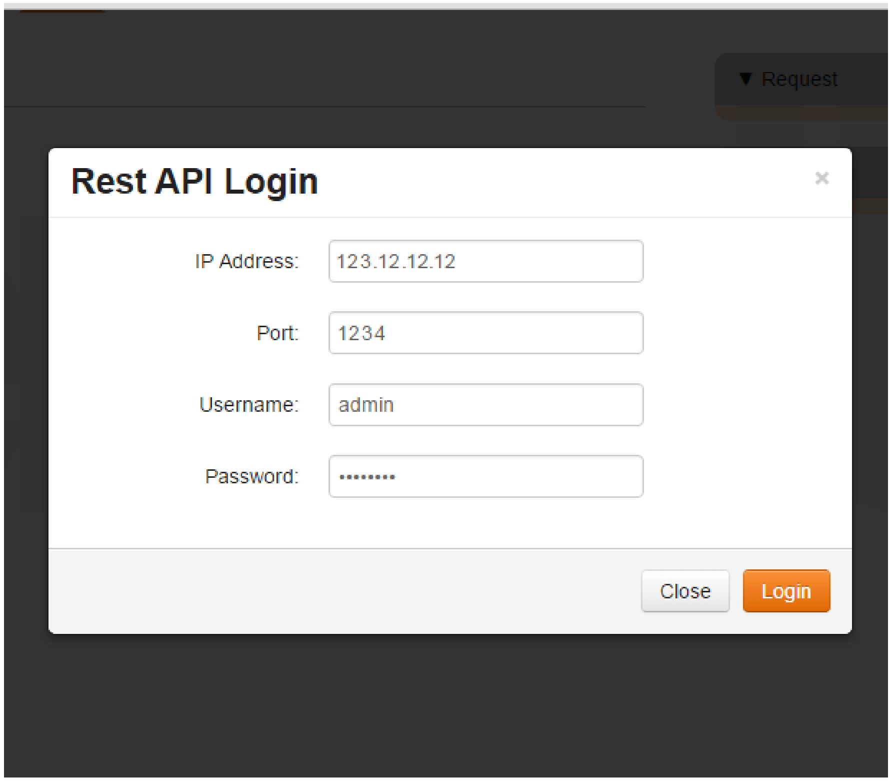
The web client window opens in a browser of your choice, or the default browser on your machine. To begin using the REST API with the web client, you must login.

Figure 1: Home Window



Login

Figure 2: Login Dialog



Rest API Login

IP Address: 123.12.12.12

Port: 1234

Username: admin

Password: ••••••••

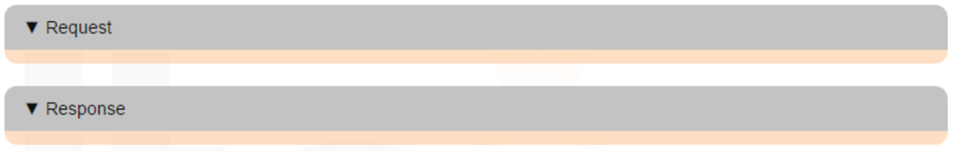
Logging into the appliance is necessary to use the REST API. The **Login** button on the home page uses password-based authentication only. Login to your HSM or partition is required only if you are authorized to use those elements. If you want to login to your HSM or partition, see "HSM" on page 40.

1. Click the **Login** button at the very top of the page. This prompts a dialog box to open.
2. Input your IP address, port number, username, and password respectively to login to the appliance.
3. Click **Login** at the bottom of the box to start working with the REST API, or **Close** to cancel your login.

User Interface

Under the **Login** button are tabs that indicate clusters of commands that the REST API can perform. Underneath each tab are buttons and prompts for information related to the commands you want to perform. Clicking on a button will run its corresponding resource automatically, and a response will be generated. Some commands run after one click, while other commands will open dialog boxes and prompts that you must populate with information.

Figure 3: Request and Response Windows



On the right are two tracking windows that show the code equivalent of your input requests and output results. The **Request** window tracks your input, and the **Response** window shows the output result. The response text turns red when a query is unsuccessful.

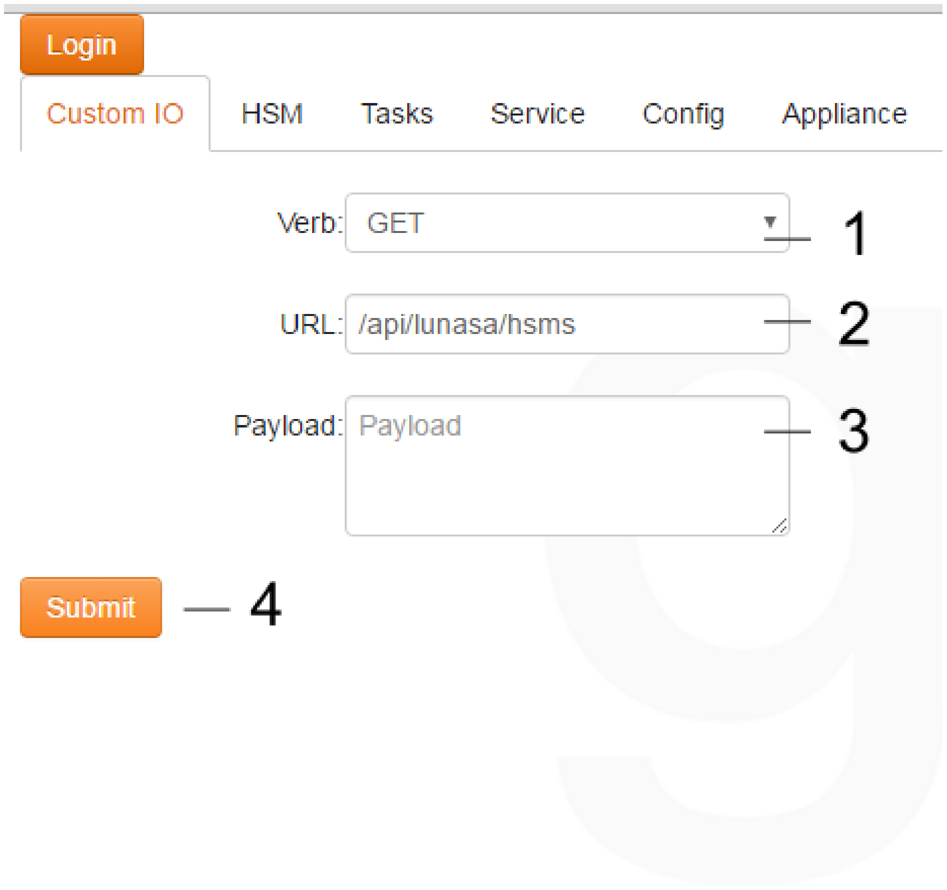
Comprehensive descriptions of each tab and its contents are contained in the following:

- "Custom IO" below
- "HSM" on page 40
- "Tasks" on page 43
- "Service" on page 44
- "Config" on page 50
- "Appliance" on page 51

Custom IO

The **Custom IO** tab allows you to input verb requests manually. Instead of clicking a button to query a resource, you can specify a resource by its verb form and request a custom payload. This tab is useful if you are familiar with the commands and do not want to switch through various tabs in the client's user interface, or if you want to query a resource not included as a button in the client.

Figure 1: Custom IO Tab



1. This drop-down menu lists verbs you can choose from.
2. The address of your query goes in this box.
3. The target payload is specified in this box.
4. After filling in items 1-3, click **Submit** to make a formal request.

Example

If you want to get all services associated with the appliance:

1. From the drop-down verb list, select **GET**.
2. Input your resource URL, **/api/lunasa/services**.
3. Edit your request using the schema in the **Payload** box to obtain your desired output.
4. Click **Submit**.

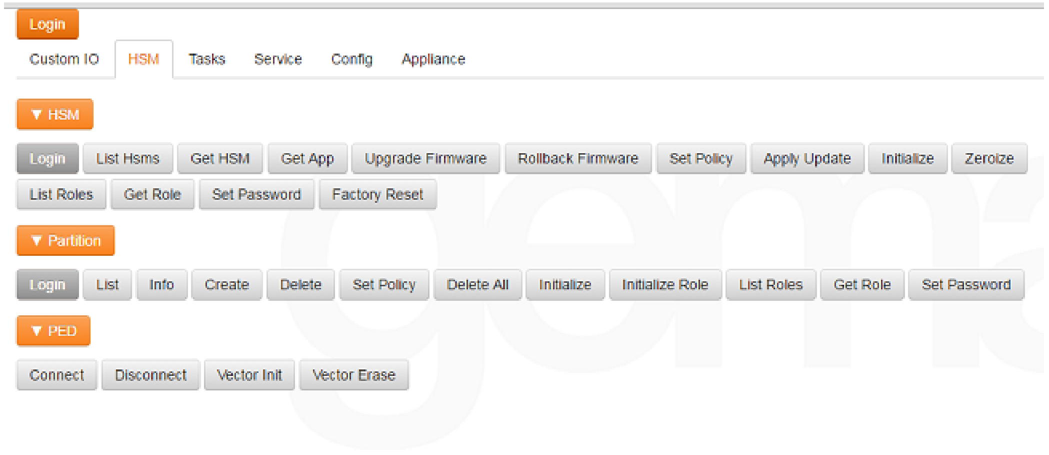
If the query is successful, you will see your request history and output in the **Request** and **Response** windows respectively.

If you are unsuccessful, the **Response** window text will turn red and return an error.

HSM

The **HSM** tab contains commands related to the HSM or partition you want to work with. Depending on how you want to authenticate, and to which element, there are different drop-down menus for each method and machine. The **HSM** and **Partition** menu **Login** buttons are used for password-based or local PED-based authentication, while the **PED** menu is specifically for connecting a remote PED. Login to the HSM or partition you want to use is required before you can use any other functions from the menus.

Figure 1: HSM Tab



The tables below list each command button you see on the **HSM** tab in the client. Each command has a short description of what it does as well as its corresponding resource. The resources can be input into the **Custom IO** tab manually if you become very familiar with them.

Note: There are some commands that are not included as buttons on the client. They must be input manually in the **Custom IO** tab. A complete repository of resources you can query with REST API can be found in the *REST API Command Reference* documentation.

Commands are grouped by the element with which you are communicating.

"HSM" above

"Partition" on page 42

"PED" on page 43

HSM

There are a few different ways to login depending on how you typically access your HSM.

- If you have direct access to your HSM and are using password-based authentication, click **Login** and input your HSM serial number and password. Click **submit**.
- If you have indirect access to your HSM via PED, click **Login** and input your user role and PED identifier number. Use 0 if you are using local PED; 1 or greater for remote PED. Click **submit**. Use the PED to complete your login.

When you are finished using REST API, or if you need to login to a different server or as a different user, click **Login** and change the HSM serial number, role, and/or password values.

The table below defines each command available under the **HSM** menu and references its corresponding resource.

Command	Function	Resource
Login	Opens a dialog box through which you can login to your HSM. Login can be done through an HSM Serial number and password, or through a role and PED depending on the type of authentication set for your device.	POST /api/lunasa/hsms/{HsmSerial}/login
List Hsms	Lists all HSMs associated with appliance.	GET /api/lunasa/hsms
Get HSM	Gets information associated with specific HSM.	GET /api/lunasa/hsms/{hsmid}
Get App	Gets information associated with the appliance .	POST /api/lunasa/hsms/{hsmid}/firmware/actions/upgrade
Upgrade Firmware	Updates HSM firmware to most recent version.	POST /api/lunasa/hsms/{hsmid}/firmware/actions/rollback
Rollback Firmware	Downgrades HSM firmware to previously installed version.	PUT /api/lunasa/hsms/{hsmid}/policies/{policyid}
Set Policy	Sets a specific HSM policy.	POST /api/lunasa/hsms/{hsmid}/updates/{updateid}
Apply Update	Applies a specific HSM update.	PUT /api/lunasa/hsms/{hsmid}/
Initialize	Initializes the HSM.	POST /api/lunasa/hsms/{hsmid}/actions/zeroize
Zeroize	• Removes all partitions and keys from the HSM. • Does not reset HSM policies, erase RPV, or delete Auditor role.	GET /api/lunasa/hsms/{hsmid}/roles
List Roles	Lists all roles associated with the HSM.	GET /api/lunasa/hsms/{hsmid}/roles/{roleid}
Get Role	Gets the information associated with a specific HSM role.	PATCH /api/lunasa/hsms/{hsmid}/roles/{roleid}
Set Password	Opens a dialog box in which you can set a new HSM password. 1. Appropriately complete the form with your new password and old password. 2. Optionally change secret and/or challenge secret associated with a particular HSM or role by changing **false** to **true** and specifying HSM serial number and role.	POST /api/lunasa/hsms/{hsmid}/actions/factoryReset
Factory Reset	Sets the HSM back to its factory default settings, deleting the HSM SO, all users, and all objects.	GET /api/lunasa/hsms

Partition

There are a few different ways to login depending on how you typically access your partition.

- If you have direct access to your partition on an HSM and are using password-based authentication, click **Login** and input your partition serial number and password. Click **submit**.
- If you have indirect access to your partition via PED, click **Login** and input your user role and PED identifier number. Use 0 if you are using local PED; 1 or greater for remote PED. Click **submit**. Use the PED to complete your login.

When you are finished using REST API, or if you need to login to a different server or as a different user, click **Login** and change the partition serial number, role, and/or password values.

The table below defines each command available under the **Partition** menu and references its corresponding resource.

Command	Function	Resource
Login	Opens a dialog box through which you can login to your HSM partition. Login can be done through a partition serial number and password, or through a role and PED depending on the type of authentication set for your device.	POST /api/lunasa/hsms/{PartSerial}/partitions/{role}/login
List	Lists all partitions associated with the HSM.	GET /api/lunasa/hsms/{hsmid}/partitions
Info	Gets information associated with a specific partition.	GET /api/lunasa/hsms/{hsmid}/partitions/{partitionid}
Create	Creates a partition.	POST /api/lunasa/hsms/{hsmid}/partitions
Delete	Removes a specific partition from the HSM.	DELETE /api/lunasa/hsms/{hsmid}/partitions/{partitionid}
Set Policy	Sets a specific partition policy.	PUT /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/policies/{policyid}
Delete All	Removes all partitions from the HSM.	DELETE /api/lunasa/hsms/{hsmid}/partitions
Initialize	Initializes the partition. *Applicable to PPSO partitions.	PUT /api/lunasa/hsms/{hsmid}/partitions/{partitionid}
Initialize Role	Initializes the partition role.	PUT /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/roles/{roleid}
List Roles	Lists all roles associated with the partition.	GET /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/roles
Get Role	Gets information associated with a specific partition role.	GET /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/roles/

Command	Function	Resource
		{roleid}
Set Password	Opens a dialog box in which you can set a new partition password. 1. Appropriately complete the form with your new password and old password 2. Optionally change secret and/or challenge secret associated with a particular HSM, partition, and/or role by changing **false** to **true** and specifying HSM serial number, partition serial number, and role.	PATCH /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/roles/{roleid}

PED

To use a remote PED for executing any tasks, you must connect to a remote PED before trying to use any of its commands.

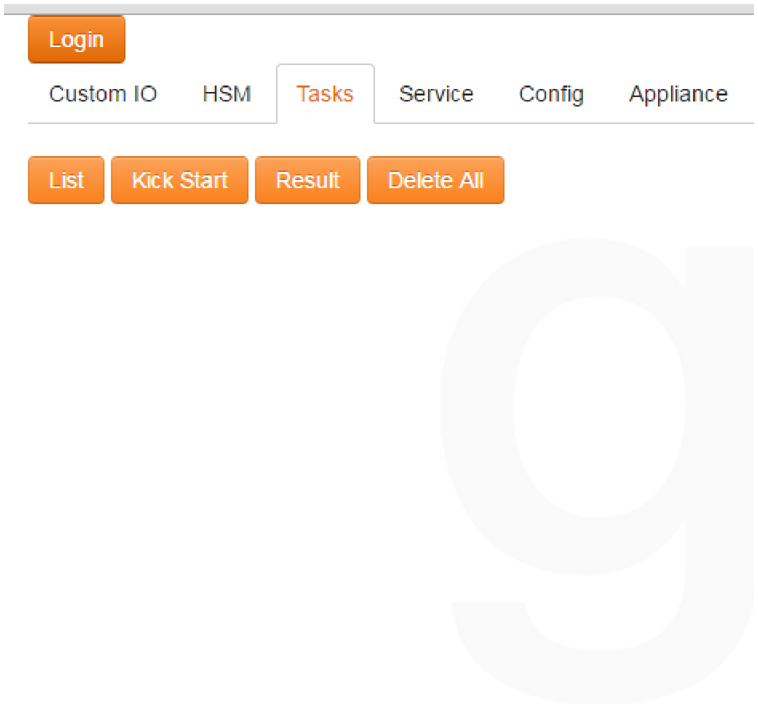
The table below defines each command available under the **PED** menu and its corresponding resource.

Command	Function	Resource
Connect	Connects to a Remote PED.	POST /api/lunasa/hsms/{hsmid}/peds/{pedid}/actions/connect
Disconnect	Disconnects the currently active Remote PED.	POST /api/lunasa/hsms/{hsmid}/peds/{pedid}/actions/disconnect
Vector Init	• Initializes a Remote PED Vector (RPV). • Creates a new Remote PED Key (RPK). • Imprints RPV onto HSM and RPK.	POST /api/lunasa/hsms/{hsmid}/peds/{pedid}/actions/vectorInitialize
Vector Erase	Erases the Remote PED vector (RPV) from the current HSM.	POST /api/lunasa/hsms/{hsmid}/peds/{pedid}/actions/vectorErase

Tasks

Tasks enable monitoring and administering of REST API resources that may require significant time to complete, such as updating HSM firmware. Rather than block and wait for these time-consuming actions to complete, REST API creates tasks to run the resource automatically in the background. Tasked resources return a response immediately and notify you of the status of the action: waiting, running, failed, etc. Because of their time-consuming nature, tasks are grouped separately.

Figure 1: Tasks Tab



The table below defines each command and references its corresponding resource.

Command	Function	Resource
List	Lists all available tasked resources.	GET /tasks
Kickstart	Starts a waiting task.	POST /tasks/{taskid}/actions/start
Result	Gets result and deletes task.	GET /tasks/{taskid}/response
Delete All	Deletes all tasks.	DELETE /tasks

Service

Services are applications that work with REST API to communicate with and manipulate information for external elements like a PC connected through a network. The **Service** tab contains actions you may perform with the services you are using.

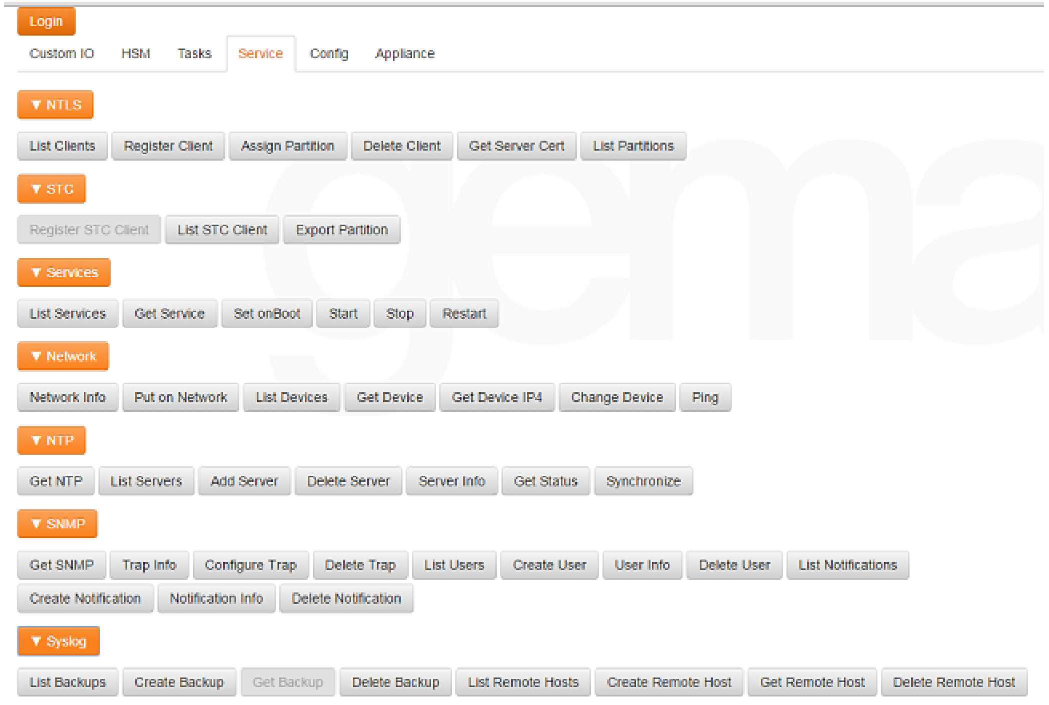
To perform a command:

1. Select which service you want to communicate with (**NTLS**, **STC**, etc.), or click the **Services** drop-down menu to send more general commands and settings.
2. Click on a command button to initiate its corresponding resource.

If the query is successful, you will see your request history and output in the **Request** and **Response** windows respectively.

If you are unsuccessful, the **Response** window text will turn red and return an error.

Figure 1: Service Tab



For example, if you want to restart a service on your partition:

1. Select the **Services** tab.
2. Click **Restart**. A dialog box will appear.
3. Populate the prompt in the dialog box with the service you wish to stop and click **submit**.

The tables below list each command button you see on the **Service** tab in the client. Each command has a short description of what it does as well as its corresponding resource. The resources can be input into the **Custom IO** tab manually if you become very familiar with them.

Note: There are some commands that are not included as buttons on the client. They must be input manually in the **Custom IO** tab. A complete repository of resources you can query with REST API can be found in the *REST API Command Reference* documentation.

The **Service** tab's drop-down menus are organized by service.

"NTLS" on the next page

"STC" on the next page

"Services" on the next page

"Network" on page 47

"NTP" on page 48

"SNMP" on page 48

"Syslog" on page 49

NTLS

NTLS, or Network Trust Link Service, guarantees a secure connection when transferring data over a network. It encrypts your data and uses two-way digital certificate authentication to protect sensitive information so that you can ensure security in your proprietary communications.

The table below defines each command available under the **NTLS** menu and lists its corresponding resource.

Command	Function	Resource
List Clients	Lists all NTLS clients registered with the appliance.	GET /api/lunasa/ntls/clients
Register Client	Registers a client with the appliance.	POST /api/lunasa/ntls/clients
Assign Partition	Registers a client with a partition on the HSM.	POST /api/lunasa/ntls/clients/{clientid}/partitions
Delete Client	Deletes a specified client from the appliance.	DELETE /api/lunasa/ntls/clients/{clientid}
Get Server Cert	Gets the server-side certificate used by NTLS to establish connections with clients.	GET /api/lunasa/ntls/certificate

STC

STC, or Secure Trusted Channel, guarantees privacy and security in user-HSM communications. STC uses encryption, message authentication codes, and bi-directional endpoint authentication to ensure that only those authorized to use the connection can do so, and that your messages remain safely protected.

The table below defines each command available under the **STC** menu and lists its corresponding resource.

Command	Function	Resource
Register STC Client	Registers a client identity for secure trusted communication with a partition.	POST /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/stc/clients
List STC Client	Lists all client identities associated with the secure trusted channel for the specified partition.	GET /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/stc/clients
Export Partition	Exports specified partition's public key to a file.	GET /api/lunasa/hsms/{hsmid}/partitions/{partitionid}/stc

Services

The **Services** menu sets general preferences for your use of any or all services with REST API.

The table below defines each command available under the **Services** menu and lists its corresponding resource.

Command	Function	Resource
List Services	Lists all services associated with the appliance.	GET /api/lunasa/services

Command	Function	Resource
Get Service	Gets specific information on the specified service.	GET /api/lunasa/services/{serviceid}
Set onBoot	Sets specified service to be running on startup.	PATCH /api/lunasa/services/{serviceid}
Start	Starts named service.	POST /api/lunasa/services/{serviceid}/actions/start
Stop	Stops named service.	POST /api/lunasa/services/{serviceid}/actions/stop
Restart	Restarts named service.	POST /api/lunasa/services/{serviceid}/actions/restart

Network

The **Network** tab manages all your network devices and your connectivity to them.

The table below defines each command available under the **Network** menu and lists its corresponding resource.

Command	Function	GET /api/lunasa/network
Network Info	Gets the network information associated with the appliance.	GET /api/lunasa/network
Put on Network	Sets all base network configurations associated with the appliance.	GET /api/lunasa/network/devices
List Devices	Lists all network devices.	GET /api/lunasa/network/devices/{deviceid}
Get Device	Gets information on the specified network device.	GET /api/lunasa/network/devices/{deviceid}/ip4
Get Device IP4	Gets IP4 information on the specified network device.	PUT /api/lunasa/network/devices/{deviceid}/ip4
Change Device	Changes network device in use.	POST /api/lunasa/network/actions/ping
Ping	Tests network connectivity to specified host.	GET /api/lunasa/network
DNS Info	Gets the DNS information associated with the network.	GET /api/lunasa/network/dns/nameServers
List Name Servers	Lists currently registered name servers.	POST /api/lunasa/network/dns/nameServers
Create Name Server	Creates a new name server.	GET /api/lunasa/network/dns/nameServers/{nameServerid}
Get Name Server	Gets information on specified name server.	DELETE /api/lunasa/network/dns/nameServers/{nameServerid}
Delete Name	Deletes name server entry.	GET /api/lunasa/network/dns/searchDomains

Command	Function	GET /api/lunasa/network
Server		
List Search Domain	Lists currently registered search domains.	POST /api/lunasa/network/dns/searchDomains
Create Search Domain	Creates a new search domain.	GET /api/lunasa/network/dns/searchDomains/ {searchDomainid}
Get Search Domain	Gets information on specified search domain.	DELETE /api/lunasa/network/dns/searchDomains/ {searchDomainid}
Delete Search Domain	Deletes search domain entry.	GET /api/lunasa/network/dns/nameServers

NTP

NTP, or Network Time Protocol, provides connections to extremely accurate servers of time data so that your appliance can be correctly synchronized. All devices can undergo gradual time drifts, and it is important to use NTP to correct these for applications to run smoothly.

The table below defines each command available under the **NTP** menu and lists its corresponding resource.

Command	Function	Resource
Get NTP	Gets NTP configuration information.	GET /api/lunasa/ntp
List Servers	Lists current server resources.	GET /api/lunasa/ntp/servers
Add Server	Adds an NTP server.	POST /api/lunasa/ntp/servers
Delete Server	Deletes all NTP server entries.	DELETE /api/lunasa/ntp/servers/ {serverid}
Server Info	Gets information on specified server.	GET /api/lunasa/ntp/servers/ {serverid}
Get Status	Returns information on ntp time, max error, estimated error, and offset.	GET /api/lunasa/ntp/status
Synchronize	Synchronizes date and time with NTP.	POST /api/lunasa/ntp/actions/synchronize

SNMP

SNMP, or Simple Network Management Protocol, monitors a local HSM for changes in certain conditions that may cause problems for users. Traps can be put in place to respond to certain condition changes and notify the appropriate personnel of errors in functioning.

The table below defines each command available under the **SNMP** menu and lists its corresponding resource.

Command	Function	Resource
Get SNMP	Gets SNMP configuration information.	GET /api/lunasa/snmp
Trap Info	Gets SNMP trap configuration information.	GET /api/lunasa/snmp/trap
Configure Trap	Configures specified SNMP trap parameters.	PUT /api/lunasa/snmp/trap
Delete Trap	Clears SNMP configuration.	DELETE /api/lunasa/snmp/trap
List Users	Lists SNMP users.	GET /api/lunasa/snmp/users
Create User	Creates an SNMP user.	POST /api/lunasa/snmp/users
User Info	Gets configuration information of specified user.	GET /api/lunasa/snmp/users/{userid}
Delete User	Deletes user.	DELETE /api/lunasa/snmp/users/{userid}
List Notifications	Lists SNMP notifications for specified user.	GET /api/lunasa/snmp/users/{userid}/notifications
Create Notification	Creates an SMP user notification.	POST /api/lunasa/snmp/users/{userid}/notifications
Notification Info	Gets configuration information for specified notification.	GET /api/lunasa/snmp/users/{userid}/notifications/ {notificationid}
Delete Notification	Deletes specified notification.	DELETE /api/lunasa/snmp/users/ {userid}/notifications/{notificationid}

Syslog

The table below defines each command available under the **Syslog** menu and lists its corresponding resource.

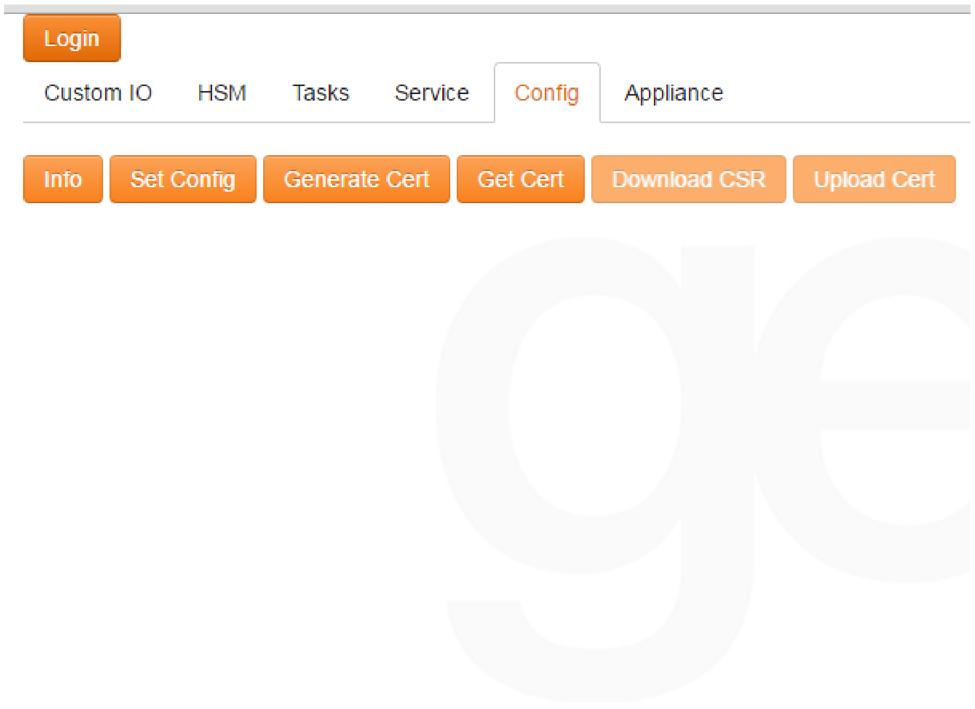
Command	Function	Resource
List Backups	Lists stored syslog backups.	GET /api/lunasa/syslog/backups
Create Backup	Creates a syslog backup.	POST /api/lunasa/syslog/backups
Get Backup	Retrieves a syslog backup and deletes it after.	GET /api/lunasa/syslog/backups/{backupid}
Delete Backup	Deletes specified syslog backup.	DELETE /api/lunasa/syslog/backups/{backupid}
List Remote Hosts	Lists configured remote hosts.	GET /api/lunasa/syslog/remoteHosts
Create Remote Host	Creates a remote host entry.	POST /api/lunasa/syslog/remoteHosts
Remote Host Info	Gets information on specified remote host.	GET /api/lunasa/syslog/remoteHosts/ {remoteHostid}

Config

The **Config** tab contains commands specifically related to the configuration of your appliance.

Note: Login to an HSM or partition is not necessary to perform operations on your appliance. However, you are limited to only these operations. Login to an HSM or partition to make full use of the REST API.

Figure 1: Config Tab



The table below defines each command available under the **Config** tab and lists its corresponding resource.

Command	Function	Resource
Info	Gets the configuration of the web server providing REST API.	GET /api/lunasa/webServer/config
Set Config	Sets the configuration of the web server providing REST API. Appropriately complete the form with the network devices you want to use, threads info, port number, key type, key size, curve name, and your list of ciphers.	PATCH /api/lunasa/webServer/config/
Generate Cert	Generates a new certificate.	GET /api/lunasa/webServer/config/certificate
Get Cert	Gets attributes of the certificate.	POST

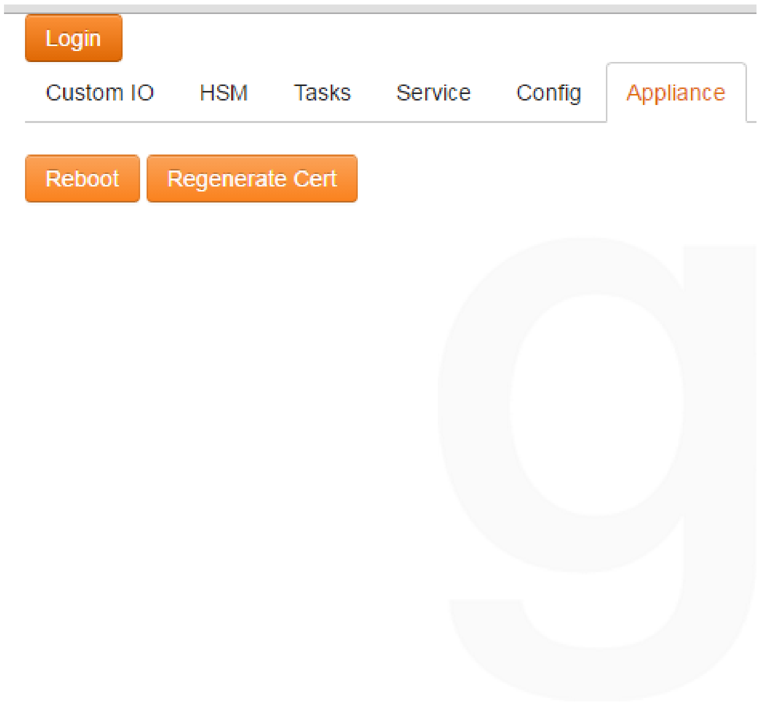
Command	Function	Resource
		/api/lunasa/webServer/config/certificate/actions/regenerate
Download CSR	Downloads a Certificate Signing Request.	GET /api/lunasa/webServer/config/csr
Upload Cert	Establishes and accepts the certificate's association with the appliance.	

Appliance

The **Appliance** tab contains basic commands related to your appliance.

Note: Login to an HSM or partition is not necessary to perform operations on your appliance. However, you are limited to only these operations. Login to an HSM or partition to make full use of the REST API.

Figure 1: Appliance Tab



The table below defines each command available under the **Appliance** tab.

Reboot	Performs a warm restart (reboot) of the appliance, shutting down all running processes in a controlled manner.
Regenerate Certificate	Deletes and replaces your certificate with a newly generated one.

3

Using the REST API

This chapter provides information on the setup of several services on the REST API. It contains the following topics:

Setting up NTLS

The steps for registering an NTLS client are described in this recipe. It is assumed that you are authenticated with the REST API.

Step 1: Generate client certificate

To generate a certificate, call **vtl createCert** with the appropriate certificate data.

Note: The common name should be something that addresses the client you wish to connect, i.e. IP or domain name.

Example:

```
c:\Program Files\SafeNet\LunaClient>vtl createCert -n 172.20.9.171
Private Key created and written to: C:\Program Files\SafeNet\Lun-
aClient\cert\client\172.20.9.171Key.pem
Certificate created and written to: C:\Program Files\SafeNet\Lun-
aClient\cert\client\172.20.9.171.pem
```

Step 2: Create client on SA

To create the client, **POST** to **/api/lunasa/ntls/clients** with the client ip, name and certificate.

Note: Encode the scripts to maintain newlines.

Example:

```
----------------Request----------------
URL:
POST:/api/lunasa/ntls/clients
Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version=3'}
Payload:
{"ip": "172.20.9.171", "clientName": "testClient", "certificate": "-----BEGIN CERTIFICATE-----
\nMIIDMKvF<.........>jlQ\nv/VBhn0=\n-----END CERTIFICATE-----"}
----------------Result----------------
Headers:
{'access-control-allow-origin': '*', 'content-type': 'application/json', 'location': '/ap-
i/lunasa/ntls/clients/testClient', 'content-length': '23', 'access-control-allow-credentials':
'true'}
Data:
{"client": "testClient"}
```

Step 3: Add server on client

To add the server to a particular client, download the server certificate, **GET** to **/api/lunasa/ntls/certificate** and save the content to a file, i.e. server.pem

Example:

```
----------------Request----------------
URL:
GET:/api/lunasa/ntls/certificate
Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version=3'}
Payload:
{}
----------------Result----------------
Headers:
{'access-control-allow-origin': '*', 'content-type': 'application/json', 'content-length':
'1202', 'access-control-allow-credentials': 'true'}
Data:
{"certificate": "-----BEGIN CERTIFICATE-----
MIIDLzCCAhegAwIBAgIBADANBgkqhkiG9w0BAQsFADBbMQswCQYDVQQGEwJDQTEQ
MA4GA1UECAwHT250YXJpbzEPMA0GA1UEBwwGT3R0YXdhMRYwFAYDVQQKDA1DaHJ5
c2FsaXMtSVRTMREwDwYDVQQDDAhoYXRlc3QyMjAeFw0xNjAyMDExNzA0MTNaFw0y
NjAyMDIxNzA0MTNaMFsxCzAJBgNVBAYTAkNBMRAwDgYDVQQIDAdPbnRhcmlvMQ8w
DQYDVQQHDAZPdHRhd2ExFjAUBgNVBAoMDUNocnlzYWxpcy1JVFMxETAPBgNVBAMM
CGhhdGVzdDIyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxZT4Jfp9
UMDP1wdUbnyGTdAmknkPXp70tAMwyUW6jFplE0+/10ipSHAxNM8J0f93xq53A3dn
k7c0hImGokwSXQgNPsJapjrbjNeFJKJP7/vFLjTu0skO0kBQa6ny8VDAnCKSq1GW
R9+GmU2TprF2DslDCs0xAsXHyRKGeZ5Og421ntFk+PX62sL6DSDzZkBO8qO2dM8x
e/R0tiIu5qltO5q5BuFcRIbkMjp00knbHBkHnYL9egRODh/4Vh8a9y48ey0OPLja
f5DgJGKQM4AXznJ8tcmsMKvPuWnQzBVGIXxvrYBawaGjfHLT0VmTnU+yOSDNZqBm
CbtrPcIsHrkfLwIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQBXDpPHmAppfLJuFHE9
1P9ALOQZa1uIYmBIZPWRlNwLIdkc4YBKCzaolqxQdmqmYpKSUm+XjDIwu+6PAvio
GwINRzI9LSB1K627Tqm6s5HPayRfsIqSkaQTsVygSCRbp0/rPy0LC+TnHvz4x0/D
aW9mc3MrQ873t2P3c8qG4oRZELQXVrmwAXA0ZI5Sc8lJFFNmVQEU5PVylKB8+cHh
xXJfkH/DHqtHjgQNYHZZlCzCPI57o5mDNq7HHXg+tpHRA7tVuRB9aj82hF22ZPA7
nByguPfmeKLUr4ETWBBJfPyuWAp2jKEzLzoKG5DFwkZAdoJ3augvvi6mCTUd/XeO
WWKM
-----END CERTIFICATE-----
"}
```

Add the server onto the client by calling **vtl addServer** with the server hostname (ip in some cases) and location to the server certificate.

Example:

```
c:\Program Files\SafeNet\LunaClient>vtl addServer -n 172.20.9.22 -c server.pem
New server 172.20.9.22 successfully added to server list.
```

Step 4: Register a partition

Assuming a partition exists on your appliance, **POST** to **/api/lunasa/ntls/clients/<clientName>/partitions** with the partition serial number, where <clientName> is the name of the client you wish to use.

Example:

```
----------------Request----------------
URL:
POST:/api/lunasa/ntls/clients/testClient/partitions
Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version=3'}
Payload:
{"partitionID": "362126088871"}
----------------Result----------------
Headers:
{'access-control-allow-origin': '*', 'content-type': 'application/json', 'location': "/ap-
i/lunasa/ntls/clients/testClient/partitions/'P1'", 'content-length': '22', 'access-control-
allow-credentials': 'true'}
Data:
{"partitionID": "'P1'"}
```

Step 5: Verify

Call **vtl verify** and check the slots for the partition.

Example:

```
c:\Program Files\SafeNet\LunaClient>vtl verify
The following Luna SA Slots/Partitions were found:
Slot Serial # Label
==== ================ =====
0 362126088871 P1
```

Setting up STC

This recipe will describe how to set up STC. It is assumed that you have already exchanged certs with the appliance. For a recipe on how to exchange certificates with the appliance see "Setting up NTLS" on page 52.

This Recipe requires both Admin API authentication and HSM API authentication.

"Step 1: Initialize client token and create identity" on the next page

"Step 2: Set HSM policy" on the next page

"Step 3: Register client" on the next page

"Step 4: Export and register partition" on the next page

"Step 5: Set partition policy" on the next page

"Step 6: Enable" on page 57

Step 1: Initialize client token and create identity

To initialize the client token, run **lunacm -q stc tki -l <clientName> -f**, where <clientName> is the name of the client you wish to use.

Example:

```
lunacm:> stc tki -l myClient -f
Successfully initialized the client token.
Command Result : No Error
```

To create the client identiy we run 'lunacm -q stc idc -l <clientName> -f', replace clientName with the one used above.

Example:

```
lunacm:> stc idc -l myClient -f
Client identity myClient successfully created and exported to file C:\Program Files\SafeNet\Lun-
aClient\data\client_identities\myClient
Command Result : No Error
```

Step 2: Set HSM policy

To set the HSM policy, **PUT** to **/api/lunasa/hsms/<HSM_Serial>/policies/39** with the value of 1.

Example:

```
----------------Request----------------
URL:
PUT:/api/lunasa/hsms/155532/policies/39
Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version=3'}
Payload:
{"value": 1}
----------------Result----------------
Headers:
{'access-control-allow-origin': '*', 'content-type': 'application/json', 'location': '/ap-
i/lunasa/hsms/155532/policies/39', 'content-length': '0', 'access-control-allow-credentials':
'true'}
Data:
""
```

Step 3: Register client

To register a client, the identity file created in Step 1 must be encoded using base64. Once complete, it may be uploaded using a **POST** to **/api/lunasa/hsms/<HSM_Serial>/partitions/<Partition_Serial>/stc/clients** including the base64 identity and label.

Example:

```
-----------------Request----------------
URL:
POST:/api/lunasa/hsms/155532/partitions/362126088871/stc/clients
Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version=3'}
Payload:
{"identity": "U2FmZU5ldFN0Y0NsaWVudElkZW<.............>FVCTElDIEtFWS0tLS0tCg==", "label":
"testClient"}
-----------------Result----------------
Headers:
{'access-control-allow-origin': '*', 'content-type': 'application/json', 'location': '/ap-
i/lunasa/hsms/155532/partitions/362126088871/stc/clients/testClient', 'content-length': '23',
'access-control-allow-credentials': 'true'}
Data:
{"client": "testClient"}
```

Step 4: Export and register partition

To export the partition, **GET** on **/api/lunasa/hsms/<HSM_Serial>/partitions/<Partition_Serial>/stc**, decode the public key and save to a file, i.e. "myPartition"

Example:

```
-----------------Request----------------
URL:
GET:/api/lunasa/hsms/155532/partitions/362126088871/stc
Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version=3'}
Payload:
{}
-----------------Result----------------
Headers:
{'access-control-allow-origin': '*', 'content-type': 'application/json', 'content-length':
'1037', 'access-control-allow-credentials': 'true'}
Data:
{"activationTimeout": 120, "clients": "/ap-
i/lunasa/hsms/155532/partitions/362126088871/stc/clients", "ciphers": "/ap-
i/lunasa/hsms/155532/partitions/362126088871/stc/ciphers", "publicKey":
"U2FmZU5ldFN0<..................................>tLS0tLQo=","fingerprint":
"81f23180aad8d29b66d8a9285ceb5638ea923984", "replayWindow": 120, "hmacs": "/ap-
i/lunasa/hsms/155532/partitions/362126088871/stc/hmacs", "rekeyThreshold": 400}
```

The next step in the process is to register the partition, run **lunacm -q stc parr -f <fileCreatedInLastStep> -l <anyName>**

Example:

```
lunacm:> stc parr -f 362126088871 -l myPartition
Partition identity 362126088871 successfully registered.
Command Result : No Error
```

Step 5: Set partition policy

To set the partition policy, **PUT** on **/api/lunasa/hsms/<HSM_Serial>/partitions/<Partition_Serial>/policies/37**

Example:

```
----------------Request----------------
URL:
PUT:/api/lunasa/hsms/155532/partitions/362126088871/policies/37
Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version=3'}
Payload:
{"value": 1}
----------------Result----------------
Headers:
{'access-control-allow-origin': '*', 'content-type': 'application/json', 'location': '/ap-
i/lunasa/hsms/155532/partitions/362126088871/policies/37', 'content-length': '0', 'access-con-
trol-allow-credentials': 'true'}
Data:
""
```

Step 6: Enable

Run **lunacm -q stc e -i 0 -f**

Example:

```
lunacm:> stc e -i 0
You are about to enable STC to server 172.20.9.22.
This will initiate an automatic restart of this application. All sessions
logged in through the application will be closed.
Are you sure you wish to continue?
Type 'proceed' to continue, or 'quit' to quit now -> proceed
Successfully enabled STC to connect to server 172.20.9.22.
Command Result : No Error
```

Setting up Public-Key Authentication

The steps for logging in with a public key via REST API are described in this recipe.

"Step 1: Create Challenge" on the next page

"Step 2: Decrypt challenge" on the next page

"Step 3: XOR" on the next page

"Step 4: Encrypt answer" on the next page

"Step 5: Answer the challenge" on the next page

Note: This assumes you have already registered a public key with the server. Instructions below if you have not.

Register public key

1. Create an RSA key pair. Construct a PEM certificate.
2. Login to server using username and password
3. Upload public key by posting to **/users/{specifiedUser}/certificates** with the certificate, where {specifiedUser} is the user you wish to use in the login process

Step 1: Create Challenge

To create a challenge, **POST** to **/api/login/challenge** with your username and your public key.

Example:

```
----------------Request----------------
URL:
POST:/auth/login/challenge
Headers:
{'Content-Type': 'application/vnd.safenetinc.lunasa+json;version=3'}
Payload:
{"username": "admin", "certificate": "-----BEGIN CERTIFICATE-----\nMDV/9........rOongA8/\n-----
END CERTIFICATE-----\n"}
----------------Result----------------
Code:
200
Headers:
{'access-control-allow-origin': '*', 'encoding': 'base64', 'content-type': 'application/json',
'content-length': '1747', 'access-control-allow-credentials': 'true'}
Data:
{"nonce": "NTdhNGVjZGQtYjhiNy00N2I5LWFmNDAtMGViYjM3MWJjMjJk", "challenge": "Hca-
ja20ca3wux..........jSQbmi3ISvf3tyFO7lKg==", "certificate": "-----BEGIN CERTIFICATE-----
\nMIIDfTC........T/AalB7Qu+i\n-----END CERTIFICATE-----\n"}
```

Step 2: Decrypt challenge

To continue with the login process, decode the challenge parameter with base64, then decrypt using the client private key.

Step 3: XOR

To get the answer to the challenge, **xor** the decoded&decrypted challenge with the base64 decoded nonce.

Step 4: Encrypt answer

To get the final challenge response, encrypt the answer with the server public key using the server certificate.

Step 5: Answer the challenge

To answer the challenge, **POST** to **/auth/login/basic** with the base64 encoded&encrypted challenge response.

Example:

```
----------------Request----------------
```

```
URL:
post:/auth/login/basic
Headers:
Payload:
{"challengeResponse": "d8RjtdS+3YjdhfU......m14YxkRAkKH8p+Wt3ZQ=="}
----------------Result----------------
Code:
204
Headers:
{'content-length': '0', 'encoding': 'base64', 'set-cookie': 'SESSION_ID=15492cf6-3c10-410e-9335-
e2e9d5ce53e0; path=/; HttpOnly', 'access-control-allow-credentials': 'true', 'access-control-
allow-origin': '*', 'content-type': 'application/json'}
Data:
""
```