GridKey OpenLV Data Centre System Overview

GK23000074

Unlocking the smartgrid

A collaboration between Lucy Electric and Sentec

Document Revision History

DATE	DESCRIPTION	REVISION
July 2017	Initial revision	0.1
August 2017	Updated to include production and pre-production differences	0.2
May 2018	Updated to include correct domain reference for API	1.0

1. Document Overview

Purpose

The purpose of this document is to provide a description of the various components that make up the OpenLV Data Centre. Each part of the Data Centre solution is presented in a separate section.

Scope

This document forms part of the Data Centre deliverable and can be supplied and used by 3rd parties in support of project methods 2 & 3. It shall also act as a reference during FAT activities.

Applicable Documents

The following are referenced in this document. Where a revision is missing, assume that the latest version applies.

REF ID	TITLE/NAME	REVISION
2383-MANUL	Public API for LV Common Application Platform.docx	V04.03.01
2626-RQSPC-SHT03	Lucy GridKey Comms Container Requirements	V00.01.03
2626-RQSPC-SHT03	Lucy GridKey Sensor Container Requirements	V00.02.01
	OpenLV Task Order	

Production and Pre-Production Data Centre Differences

This document was formed with the Pre-Production Data Centre as a reference. From a functional system point of view the 2 setups are identical. However, the Production Data Centre has additional logging capabilities, backups, higher availability of the database nodes and more importantly a different domain name. Therefore, all domain names presented in the examples and supporting information are referencing the Pre-Production system domain **test.gridkey.uk**.

For the production system this domain will change to **opalmatrix.gridkey.uk**, to reflect the anonymised project name **opalmatrix.**

Therefore, for the purposes of mapping across to the Production Data Centre, the following changes apply:

1. All references in this document to the API https://test.gridkey.uk shall be interpreted as https://api.opalmatrix.gridkey.uk
2. All references in this document to the graphing web server https://graph.test.gridkey.uk shall be interpreted as https://graph.opalmatrix.gridkey.uk

2. System Overview

The system consists of multiple EC2 instances split across two network ACL groups, as shown in the Figure 1 below:

- Secure LAN
  - Has no direct access to the internet. Only the app server has internet connectivity through the web server acting as a proxy.
  - Database nodes have a consistency level of 2 and therefore replicate data.
  - Data from database nodes are backed up to an S3 bucket. A full backup takes place weekly with hourly incremental backups.
- DMZ
  - Contains a server which monitors instance health (CPU, RAM, Hard disk space etc.)
  - Contains a logging server which logs all inbound and outbound traffic. All traffic passes through this node.
  - Contains a web server. This web server hosts an apache server to display webpages to the end user. Apache provides proxies for the API from the app server to the end user. This server also acts as a proxy between the Open LV units and the app server to add an additional layer of security to the secure LAN.
  - Contains a VPN bastion host to access SSH and other locked down services to administer servers.

All servers have write access to S3 to enable logging and backups to be stored securely offsite. This ensures security of backups and integrity of firewall logging data and rsyslog data.

SES is available from all servers to enable emails to be sent should there be a need to alert an administrator.

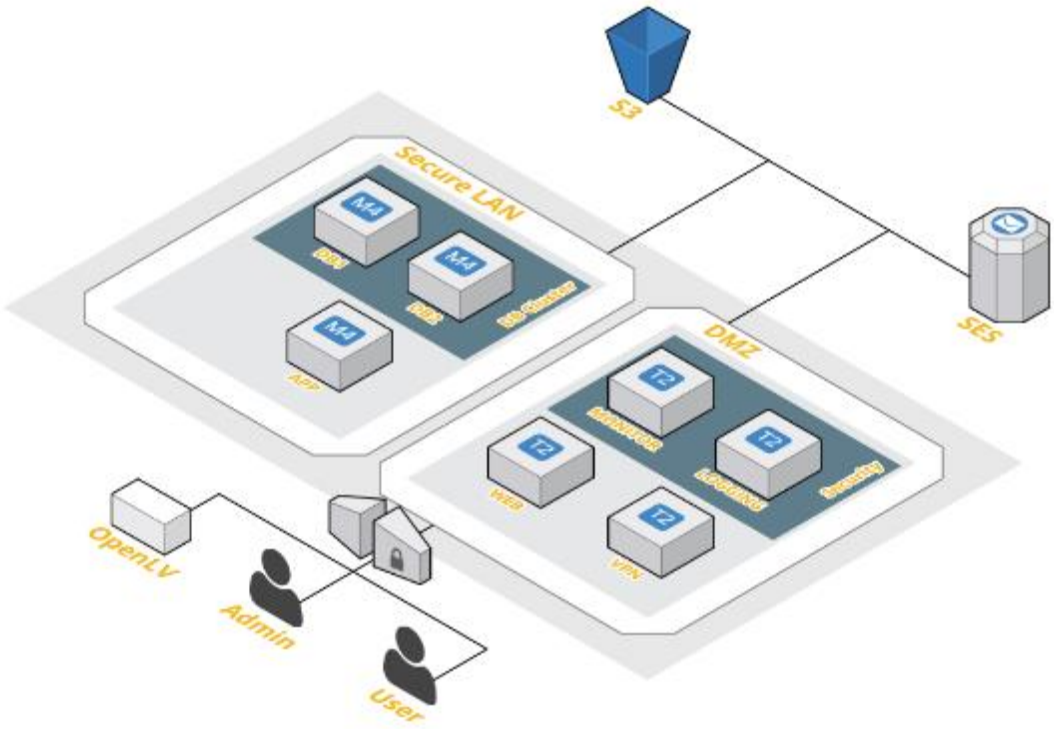

Figure 1 - AWS System Overview

3. Data Processing/Storage

Directory Scanning

The OpenLV Communications Container performs an upload of data via SFTP to the remote Data Centre. These files are then parsed by another application that runs on the server that polls a specific directory before processing the file and committing the contents to the database.

Once a file has been parsed it is then stored on the file system in a temporary archive directory. The purpose of this archive directory storage is to allow for validation of the uploaded data as part of FAT activities, or for debug purposes should the transfer of data or its contents need to be verified.

Files that are uploaded should adhere to the following naming format:

<Unit Serial Number> - <Time Uploaded Unix Timestamp>.json

e.g.

OpenLV-06 – 1505796823.json.

Files that do not adhere to this format may not be parsed correctly as the unit serial number is derived from the filename as it is not present in the file content, and it is required to be able to uniquely index the data in the database.

Backlog Processing

When a communications outage occurs, a situation could arise where a backlog of data is pending transmission. To avoid problems with Unix timestamps only having ‘second’ resolution, files that are uploaded in the same second have a monotonically increasing sequence number appended to the file name. See example in Figure 2 below:

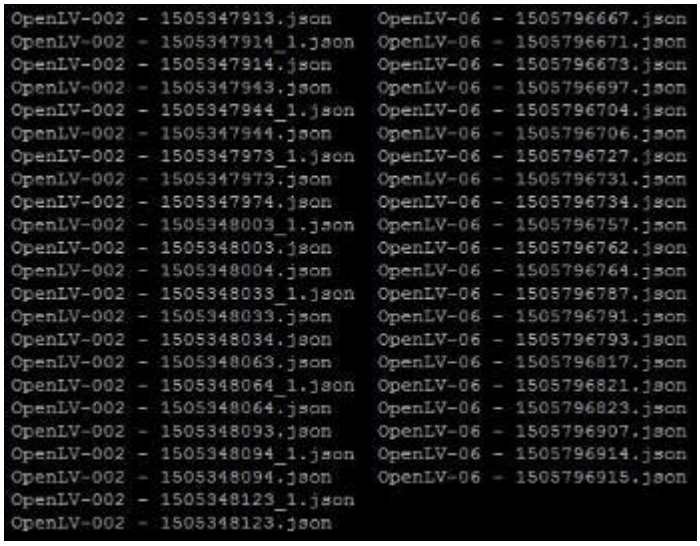

Figure 2 - Example Backlog File Naming

Database Schema

The database schema has been designed so that there is a logical breakdown of data, to ensure colocation and ensure that queries execute in a timely manner. As such, data is partitioned by the OpenLV serial number (as configured in the Communications Container configuration file and therefore present in the uploaded filename), and clustered based upon the data Instance ID. Therefore, it is essential that OpenLV serial numbers are unique to avoid each units’ data overwriting one another.

When processing files, the contents are validated to ensure that they parse as valid JSON, and contain the various key fields specified as required in *2383-MANUL*.

Data Storage

The file content is stored in the database in 2 forms. The first is “raw” and the second is “decoded”.

When data is stored in its "raw" form, it is essentially stored as the JSON string, unprocessed, as it was uploaded from the OpenLV platform i.e. this is essentially a storage/response/newdata/<IID> topic. In this way, data can be replayed in the system should that be necessary, or bulk exported on a day by day basis more easily. When storing this data, the “Timestamp” field in the payload is used as the time reference. Where this is omitted the file upload timestamp, extracted from the filename, is used instead.

The second form of data storage depends upon the JSON Object Structure type.

Data for Scalar JSON Object Structures are stored in their “decoded” form. This is so that data can be more easily plotted without first having to be further processed. This is achieved by extracting the “Timestamp” and “Value” fields from the parent “Data” field.

Data for non-scalar JSON Object Structures are also stored as “raw” individual messages. This is achieved by taking each table row in the "storage/response/newdata/IID" topic and storing the "Data" field against its "SubTopic" name field. The "Data" field is not processed or further decoded.

The reasoning behind this is because these fields are not decoded in the same way as scalar object types, as they do not lend themselves to being plotted etc. Instead they lend themselves to being extracted using the API and processed programmatically by a 3rd party application that better understands the context of the data.

4. API

As part of Methods 2 & 3 there is a requirement to be able to extract data in a usable format. Therefore, a RESTful API has been provided to meet this need. As part of this, 3 basic API calls are available as documented in the rest of this section.

Common to each API call is the need to ensure that the appropriate authorisation is supplied. In most browsers, the ability to pass in via the URL has been removed. Therefore a 3rd party extension may need to be installed – suggest ModHeader if using Chrome.

The header information needed by the ModHeader extension is “Authorization” with a value of:

"Basic Y2MzODYyYmUtNGU4Yy00NWZmLWFhODgtMGJkYzAwYjA1MDM4OjQ0NzFjNWI1LWJmMjctNGU2NS1iZWI3LTYwOGMxYzhjMmI5ZA==" – note this is an example of the credentials that would supplied and not valid on the production system.

The following, Figure 3, shows the ModHeader extension with the appropriate information supplied (albeit truncated)

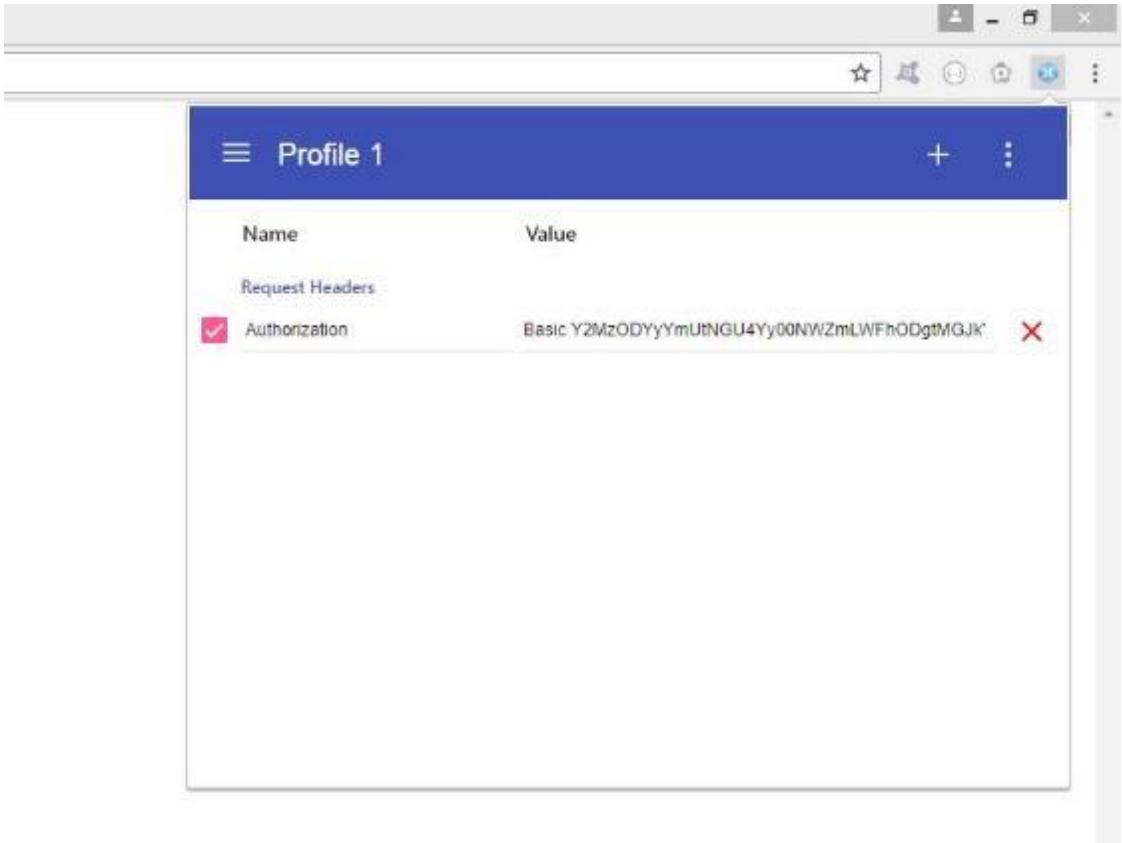

Figure 3 - Mod Header Authorisation Example

For methods 2 and 3, If a third party implements server-side code, they could hard code this authorization header into their HTTP GET requests and the end user wouldn't be required to enter it manually. They would also be supplied with an authorisation set that is unique to them, and that provides an appropriate level of access i.e. only certain OpenLV platforms or IID’s information would be accessible.

List Keyspaces

The first lists the available Cassandra Keyspaces available. This is of particular interest for the FAT, for the purpose of demonstrating segregation of the OpenLV Data Centre from the business-as-usual GridKey Data Centre.

The URL is https://test.gridkey.uk/v1

Which produces the output shown in Figure 4:

Figure 4 - Display Keyspaces

This is the entire set of Keyspaces in the system. The presence of other Keyspaces in the list would imply the non-segregation of data.

Retrieve Raw Data

The second API call is associated with retrieving the “raw” data as described in Section 3 Data Storage in this document. This allows for both the originally received unprocessed file, and also non-scalar JSON Object Structures to be retrieved based upon specific subtopic names.

The URL is https://test.gridkey.uk/v1/OPENLV01/{unit-sn}/{iid}/raw/{date}/{topic}:

Where:

- {unit-sn} is the Unit Serial Number as defined in the Communications Container configuration file and will be unique for each OpenLV hardware instance
- {iid} The IID of the container application that the data originated from
- {date} Data is requested for “raw” data a day at a time at most. This is to prevent large amounts of data being returned or the query potentially timing out. The format is YYMMDDhhmmss. Currently on the YYMMDD parts are used, the hhmmss fields should be set to 0 (this is a placeholder for later expansion if required).
- {topic} This is the topic name, with any “/” replaced with “-“. This is set to “response” to retrieve the data contained in the files that were uploaded. For other topics that are of non-scalar JSON Object Structures, use the sub topic name e.g. “bar-voltage-predict-byday-T-30-30-720”. This method of replacement still functions correctly if topic names already contain the “-“ character.

Note that due to the JSON parser used, Long and Integer data types are cast to Doubles. Therefore, for retrieval of raw data that is not “response” i.e. a sub topic, the timestamp fields will be retrieved in scientific notation format.

Whilst the full original value has been preserved any 3rd party applications that attempt to parse this data to a Date/Time library will need to cast back to Long/Integer type.

It is expected that this API call would be most useful for the purposes of replaying data back through the system, should any analytics need to be re-run on the data (not currently in scope), or for bulk export of data one day at a time.

An example output is as shown in Figure 5:

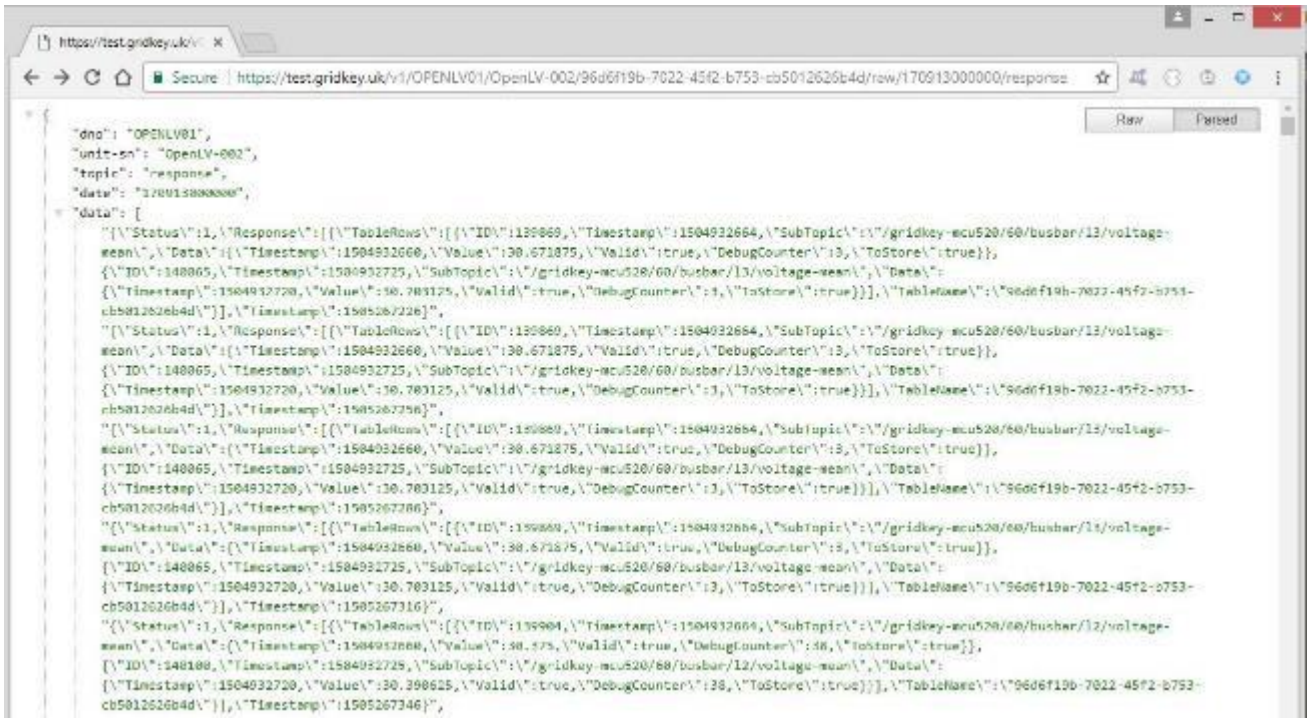

Figure 5 - Raw API Call

Retrieve Decoded Data

The third API call is associated with retrieving the “decoded” data as described in Section 3 Data Storage in this document. It is therefore expected that this API call will only be used for the extraction of scalar JSON Object Structures.

The URL is https://test.gridkey.uk/v1/OPENLV01/{unit-sn}/{iid}/decoded/{start}/{end}/{topics}:

Where:

- {unit-sn} is the Unit Serial Number as defined in the Communications Container configuration file and will be unique for each OpenLV hardware instance
- {iid} The IID of the container application that the data originated from
- {start} Start date in the form of YYMMDDhhmmss; the hhmmss fields should be set to 0 (this is a placeholder for later expansion if required).
- {end} End date in the form of YYMMDDhhmmss; the hhmmss fields should be set to 0 (this is a placeholder for later expansion if required).
- {topics} This is a comma separated list of topic names with any “/” replaced with “-“. This method of replacement still functions correctly if topic names already contain the “-“ character.

Whilst multiple topics can be requested (unlike raw where only one can be specified) a large number of topics will increase the response time of the API call potentially leading to a timeout.

Figure 6 is an example of a decoded API call:

Figure 6 - Decoded Data API Call

5. Demonstration Web Server

For the purpose of providing a demonstration of visualising data stored in the Cassandra database, a public facing site has been provided. This allows for various parameters to be plotted for a specified time period. As this page was constructed for the purpose of demonstration during the FAT, the range of parameters available for display will be limited.

The URL is https://graph.test.gridkey.uk

An example of the graph output is shown in Figure 7.

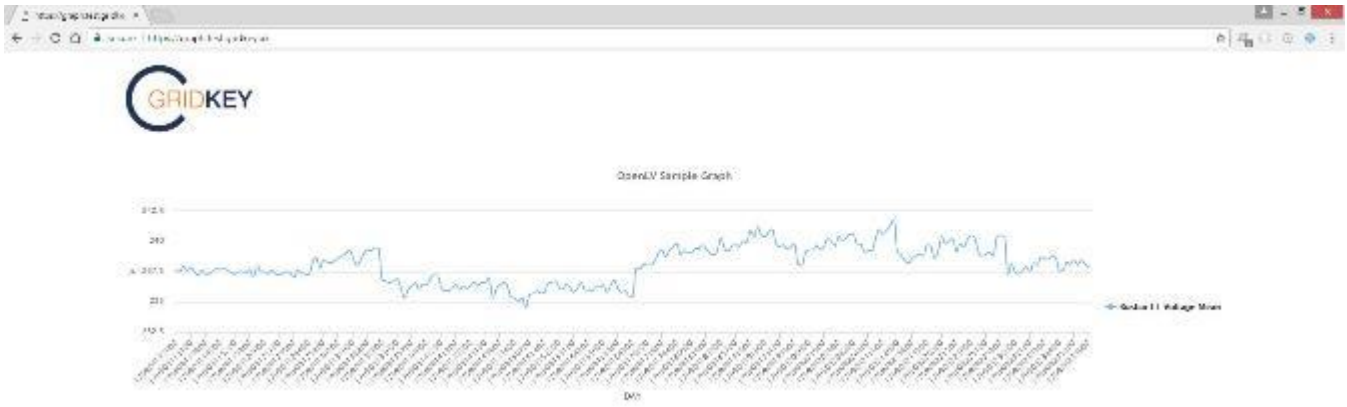

Figure 7 - Example Graph Using API