KeyStore – Rendezvous Server

HTTP API v1

[Introduction 2](#_Toc512793085)

[Migration 2](#_Toc512793086)

[Change Log 2](#_Toc512793087)

[Version 1 (2018-04-29) 2](#_Toc512793088)

[Authorization/Authentication 2](#_Toc512793089)

[Dates & Times 2](#_Toc512793090)

[Limits 3](#_Toc512793091)

[Errors 3](#_Toc512793092)

[Statuses 3](#_Toc512793093)

[Body JSON Structure 3](#_Toc512793094)

[Common 4](#_Toc512793095)

[Endpoints 5](#_Toc512793096)

[POST /keystore 5](#_Toc512793097)

[DELETE /keystore/one 5](#_Toc512793098)

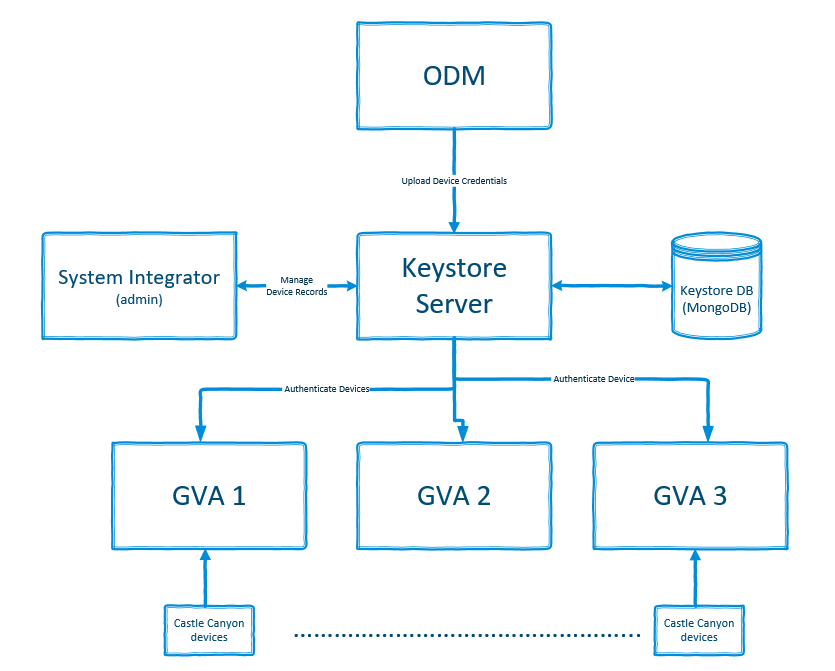
[DELETE /keystore/all 5](#_Toc512793099)

[DELETE /keystore/authenticate 6](#_Toc512793100)

# Introduction

As Keystore API will support –

1. ODMs to upload device credentials.
2. SI to manage the keystore
3. GVA to authenticate the devices



As Rendavouzs server:

1. Server can act a point of contact for CC devices,, (call home server).
2. Store iot hub credentials for GW devices,
3. Store forwarding GVA address, credentials, Shipping IDs etc..



## Migration

This API will require a shift from a direct application-to-database topology to an application-to-web-service topology. Instead of tight coupling to a specific database solution and direct access to the data's JSON representation, the database solution will be abstracted by design. Specifically, while JSON may be accepted by a given endpoint, the data is not necessarily stored in that type of structure. Design discussions will shift from centering on monolithic database schemas to HTTP endpoints and the data fragments/shapes they expose based on need.

An effort has been made to reduce the migration scope through means such as retaining as many field name spellings as possible and minimizing the endpoints.

# Change Log

## Version 1 (2018-04-29)

Initial version with:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | API | Role | Description |
| POST | /keystore | ODM/SI | Add new devices to the keystore |
| DELETE | /keystore/one | SI | Delete a device record from the keystore |
| DELETE | /keystore/all | SI | Delete all devices records from the keystore |
| GET | /keystore/authenticate | GVA | Authenticate a Castle Canyon Gateway or Tag device |

## Version 2 (2018-06-12)

Update database format.

Provided Add records with Curl command

# Authorization/Authentication

In this revision the keystore does not support any Authorization or Authentication

# Dates & Times

All dates & times, both sent to the keystore and received from the GVA, should be UTC.

In order to support GVA clients in any time zone, each client will need to convert to UTC when sending a time.

# Limits

* Request body size: 5 megabytes

# Errors

## Statuses

This API's error handling responses will focus on these status codes.

* 400
  + If any required JSON element is missing/empty or basic validation checks (e.g. two entity counts are unequal but expected to be equal) fails.
  + If the request attempts to create an entity that already exists, e.g. an ID conflict was detected.
  + If credentials were provided but incomplete.
* 401
  + If credentials were not provided.
* 403
  + If the credentials are otherwise valid but the role associated with them was not granted access to this action.
* 404
  + If the identified resource does not exist (or is not visible to the user based on access grants).
* 409
  + If the request cannot be fulfilled because doing so would conflict with the current state of the resource.
* 500
  + If the GVA fails to process the request for any reason not covered by the above codes.

## Body JSON Structure

* **"requestId"**: <string>
  + Request ID to assist correlation between bug reports and log inspection.
* **"status"**: <number>
  + HTTP status code
* **"code"**: "machine\_readable\_string"
  + Provides application-level codes (versus HTTP codes) to describe the error in more detail and also provide stable strings for conditional logic (as an alternative to regex on human-readable messages).
* **"message"**: "human readable string which is free to change at any time"
  + Provides client developers more information in prose, with the freedom to include things such as URLs to issues/documentation/etc. and also the freedom to change the content at any time.
* **"detail"**: <object>
  + Provides either an empty object or one of arbitrary (likely shallow) depth with additional itemized/organized information to augment the code and message, e.g. names of fields received in the request (as the keys) and their validation errors).

## Common

For each of these codes, the accompanying message/detail fields in the JSON should provide more detail about the specific scenario.

#### bad\_request

This code will be returned in HTTP 400 scenarios.

#### unauthorized

This code will be returned in HTTP 401 scenarios.

#### forbidden

This code will be returned in HTTP 403 scenarios.

#### server\_error

This code will be returned in HTTP 5XX scenarios.

#### input\_validation\_failed

This code will be returned in HTTP 400 scenarios such as:

* Missing/empty fields whose values are required.
* Values with unacceptable formats, lengths, etc. based on criteria such as business rules or database schema constraints.

# Endpoints

## POST /keystore

* Add a new device record to the keystore database.
* Response
  + Success
    - Status: 200
  + Errors (beyond those in the "Errors > Common" section)
    - Status: 202 along with details in the body(result).

|  |
| --- |
| Request Structure (body) |
| {      "deviceUuid": String,    "typeOfDevice": String,    "hwRevision": String,    "swRevision": String,    "odmId": String,    "ownerName": String,    "gvaId": String,    "dateOfManufacture": String,    "signedEcdsaPublicKeyCertificate": String,    "deviceEcdhPublicKey": String,    "creationDate": current time in ms } |

|  |
| --- |
| Response Structure |
| {  "result": String  } |

## GET /keystore

## Get the keystore record for a particular device UUID.

* Request
  + Query Arguments
    - UUID: UUID of the device for which record
* Response
  + Success
    - Status: 200
  + Errors (beyond those in the "Errors > Common" section)
    - Status: 202 with reason in Body.

|  |
| --- |
| Response Structure (body) |
| {      "deviceUuid": String,    "typeOfDevice": String,    "hwRevision": String,    "swRevision": String,    "odmId": String,    "ownerName": String,    "gvaId": String,    "dateOfManufacture": String,    "signedEcdsaPublicKeyCertificate": String,    "deviceEcdhPublicKey": String,    "creationDate": time in ms } |

## POST /keystore/multiple

* Add multiple records at once from a csv file using curl command
* Curl command to upload data:
  + curl –X POST –F “data=@filename.csv” http://ipaddress:portnumber/keystore/multiple
* Format of csv file, Coloumns
  + typeOfDevice
  + hwRevision
  + swRevision
  + odmId
  + ownerName
  + gvaId
  + dateOfManufacturer
  + signedEcdsaPublicKeyCertificate
  + deviceEcdhPublicKey
  + creationDate
* Response
  + Success
    - Status: 200
    - Body: None.
  + Errors (beyond those in the "Errors > Common" section)
    - Status: 202 reason is in body.

## PUT /keystore/update

* Update a parameter value in database
* Request
  + Query Arguments
    - UUID: UUID of the device
    - parameter: Parameter whose value is to be updated
    - newparamvalue: New value of the parameter
* Response
  + Success
    - Status: 200
    - Body: None.
  + Errors (beyond those in the "Errors > Common" section)
    - Status: 202 reason is in body.

|  |
| --- |
| Response Structure (body) |
| {    } |

## DELETE /keystore/one

* Delete a single record from the keystore.
* Request
  + Query Arguments
    - UUID: the UUID of device that needs to be deleted
* Response
  + Success
    - Status: 200
    - Body: None.
  + Errors (beyond those in the "Errors > Common" section)
    - Status 202, details in the body.
* Idempotent
  + Yes.

## DELETE /keystore/all

* Delete a all record from the keystore.
* Request
  + Query Arguments
    - password: added for safety.. value needs to be password=IwantToDeleteAll
* Response
  + Success
    - Status: 200
    - Body: None.
  + Errors (beyond those in the "Errors > Common" section)
    - Status: 202, details in the body.
* Idempotent
  + Yes.

## POST /keystore/authenticate

* Delete a all record from the keystore.
* Request
  + Query Arguments
    - UUID: UUID of the device that needs to be authenticated
    - Challenge: Challenge string provided to GW for authentication
    - Response: Challenge response sent from GW
* Response
  + Success
    - Status: 200
  + Errors (beyond those in the "Errors > Common" section)
    - Status: 202 with reason in the response body.
* Idempotent
  + Yes.

|  |
| --- |
| Response Structure |
| {      "deviceUuid": String,    "typeOfDevice": String,    "hwRevision": String,    "swRevision": String,    "odmId": String,    "ownerName": String,    "gvaId": String,    "dateOfManufacture": String,    "signedEcdsaPublicKeyCertificate": String,    "deviceEcdhPublicKey": String,    "creationDate": String } |

## POST /register

* Purpose
  + Register given gateway device to IoT Hub and return IoT Hub / shipping API connection credentials along with associated shipment ID.
* Authorized Role(s)
  + Dock Worker (SI)
* Request
  + Query Arguments
    - gatewayUuid: UUID of the device that needs to be registered
      * Validation
        + Valid 32 Bit UUID.
        + Not Null
* Response
  + Success
    - Status: 201
    - Body: mentioned below.
  + Errors (beyond those in the "Errors > Common" section)
    - Status: 204 with reason in the response body.
* Idempotent
  + No
* Notes:
  + Need to have mechanism to query for multiple GVA instances and fetch appropriate shipment id associated with the current gateway (Simulate Rendezvous Server functionality).

|  |
| --- |
| Request Structure (body) |
| {      "gatewayUuid": <String: gateway ID> } |

|  |
| --- |
| Response Structure |
| {  "gatewayUuid": <String: gateway ID>,  "connectionCredentials": {  "shippingApi": {  "shipmentId": <String>  "shippingApiUrl": <String>,  "username": <String>,  "password": <String>,  "role": <String>,  "token": <String : Token>,  },  "deviceCredentials": {  "cloudType": <cloud Type, eg: azure, gcp String>,  "azure": {  "uri": <String>,  "deviceId": <String>,  "primaryKey": <String>,  "seconderyKey": <String>,  "connString": <String>  },  "gcp": {  "registryId": <String>,  "projectId": <String>,  "rsaPrivate": <String>,  "rsaPublic": <String>,  "rsaCert": <String>  }  }  }  } |

## POST /keystore/updateDeviceRecord

* Purpose
  + Called by GVA (non-Obt flow), to update device credentials with the GVA the device is associated with, and iot hub credentials.
* Request
* Response
  + Success
    - Status: 201
    - Body: mentioned below.
  + Errors (beyond those in the "Errors > Common" section)
    - Status: 204 with reason in the response body.
* Idempotent
  + No
* Notes:

|  |
| --- |
| Request Structure (body) |
| {  "deviceId": "simulated007",  "shipmentId": "325",  "gvaId": "gva.ind.com",  "connectionString": {  "cloudType": <cloud Type, eg: azure, gcp String>,  "azure": {  "uri": <String>,  "deviceId": <String>,  "primaryKey": <String>,  "seconderyKey": <String>,  "connString": <String>  }  }  } |

|  |
| --- |
| Response Structure |
|  |

# Postman Script v1.

