GVA Security APIs v1

[Introduction 2](#_Toc513119738)

[Authorization/Authentication 2](#_Toc513119739)

[Dates & Times 2](#_Toc513119740)

[Limits 2](#_Toc513119741)

[Errors 2](#_Toc513119742)

[Statuses 2](#_Toc513119743)

[Body JSON Structure 2](#_Toc513119744)

[Common 3](#_Toc513119745)

[Endpoints 4](#_Toc513119746)

[POST /authenticate/:id (new GVA API) 4](#_Toc513119747)

[POST /shipment/:id/monitor (GVA api already exists) 5](#_Toc513119748)

# Introduction

List of End points to be added to GVA as part of security implementation.

# Authorization/Authentication

# Dates & Times

# 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 /authenticate/:id (new GVA API)

* Authenticate the GW and create IOT HUB keys
* Response
  + Success
    - Status: 200
  + Errors (beyond those in the "Errors > Common" section)
    - None.

|  |
| --- |
| Request Structure |
| {  "deviceUuid": "<STRING: UUID of the Gateway>",  "gwEcdhPublicKey": "<Gateway’s ECDH Public key",  “obtEcdsaPublicKey”: “<OBT’s ECDSA Public Key>”  } |

|  |
| --- |
| Response Structure |
| {  “sessionCredentials”: {  “header”:{  "signingAlgorithm": "ECDSA",  "hashMethod": "sha256",  "signatureEncoding": "base64",  "signedBy": "GVA",  "signature": "<base64: Signature of HASH of payload>"  },  "payload": "{  "gvaEcdsaPublicKey": "<GVA’s ECDSA public key>",  "gvaEcdhPublicKey": "<GVA’s ECDH Public key>",  "sessionNonce": "<16byte random string>",  "shipmentId": <Integer>,  "obtEcdsaPublicKey": "<OBT’s ECDSA Public Key>"  }"  },  "gwConnectionCredentials": {  "header":{  "signingAlgorithm": "ECDSA",  "hashMethod": "sha256",  "signatureEncoding": "base64",  "signedBy": "GVA",  "signature": "<base64: Signature of HASH of payload>"  },  "payload": "<base64: encrypted(Ek) of gwConnectionCredentials structure>"  }  } |

|  |
| --- |
| gwConnectionCredentials: |
| {  "commonBeaconKey": "<temporary Beacon secret for WSN>",  "iotConnectionType": "IOTHUB",  "connectionProperties":{  "iotHubUri": "<Azure IOT HUB’s URI>",  "primaryKey": "<primary key for GW device>",  "secondaryKey": "<secondary key for GW device>",  }  } |

Reference structure:

## POST /shipment/:id/monitor (GVA api already exists)

* Transition the shipment’s status.
* Authorized Role(s)
  + Dock Worker
* Shipment Status Transition
  + From "inProvision" to "inMonitoring"
* Shipment Status Lock
  + OBT user releases the lock.
* "Tag 0" support
  + For scenarios where the gateway itself is acting as a tag, it will be represented as a tag object under one of the "shippingUnit" nodes. The OBT will set the tag's "id" to the gateway's UUID and the tag's "wsnId" to the gateway's WSN ID.
* Response
  + Success
    - Status: 204
    - Body: None.
  + Errors (beyond those in the "Errors > Common" section)
    - status\_transition\_forbidden
    - status\_transition\_invalid
* Idempotent
  + Yes. GVA will avoid side-effects if the shipment is already in the requested state. GVA will return the same endpoint-standard success response.
* Reference Points
  + propertyStatus property overview.docx (row 4)

|  |
| --- |
| Request Structure |
| {  "gateways": [  {  "id": "<UUID>",  "challengeResponse": "<base64 byte length>",  "ecdhPublicKey": "<base64 byte length>"  "shippingUnits": [  {  "id": "<tracking ID>",  "tags": [  {  "id": "<UUID>",  "wsnId": <integer>,  "challengeResponse": "<base64 byte length>",  "ecdhPublicKey": "<base64 byte length>"  }  ]  }  ]  }  ]  } |

|  |
| --- |
| Response Structure |
| {  "header":{  "signingAlgorithm": "ECDSA",  "hashMethod": "SHA256",  "signatureEncoding": "base64",  "signedBy": "GVA",  "signature": "<base64: Signature of HASH of payload>"  },  "payload": {  "beaconSecrets": [  {  "uuid": "<String: UUID of GW or Tag device>",  "encryptedBeaconSecret": "<base64: byte length>"  }  ]  }  } |