	rfxcel Traceability Hub (rTH)	rTH-Design-Spec-09
	Design Specification	Page 1 of 17




RFXCEL
ANTARES VISION GROUP

POS rTH REST API v1.0

Design Specifications


Table of Contents

- 1 DOCUMENT CONTROL.....3**
- 2 DOCUMENT CHANGE HISTORY4**
 - 2.1 CHANGE HISTORY 4
- 3 INTRODUCTION5**
 - 3.1 PURPOSE..... 5
 - 3.2 REFERENCE DOCUMENTS..... 5
 - 3.3 ABBREVIATIONS AND ACRONYMS 5
- 4 GENERAL DESCRIPTION5**
 - 4.1 BACKGROUND 5
 - 4.2 FUNCTIONAL REQUIREMENTS 6
- 5 INTERFACES.....6**
 - 5.1 COMMON AUTHORIZATION HEADERS 6
 - 5.2 OAUTH 2.0 AUTHORIZATION 7
 - 5.3 COMMON HTTP RESPONSE CODES..... 8
- 6 STATUS CHECK API8**
- 7 CREATE EVENTS- JSON FORMAT13**
 - 1. *Create Dispense Event*..... 14
 - 2. *Create Return Event* 15
 - 7.1 GET EVENT CREATION RESULT FROM ASYNCHRONOUS PROCESSING 16

	rfxcel Traceability Hub (rTH)	rTH-Design-Spec-09
	Design Specification	Page 3 of 17

1 DOCUMENT CONTROL

Approval Sign Off

	rfxcel Traceability Hub (rTH)	rTH-Design-Spec-09
	Design Specification	Page 4 of 17

2 Document Change History

2.1 Change history

Version	Date	Changes
1.0	07-22-2022	Initial version of the rTH POS API documentation

3 INTRODUCTION

3.1 Purpose

The purpose of this document is to describe the Point-of-Sale (POS) REST API of the rTH system.

3.2 Reference Documents

NA

3.3 Abbreviations and Acronyms

Abbreviation	Description
TBD	To Be Defined
SFS	Software Functional Specification
NA	Not Applicable
EPCIS	Electronic Product Code Information Services
REST	Representational state transfer
API	Application Programming Interface
rTH	rfxcel Traceability Hub
POS	Point of Sale

4 GENERAL DESCRIPTION

4.1 Background

This document provides the specification for rTH POS REST APIs used to check product status and create Dispense/Return events.

4.2 Functional Requirements

- FUNCTIONAL REQUIREMENTS
 - REST APIs will return results based on appropriate input variables.
 - REST APIs to create event data supporting JSON formats.
 - Authorization is required for each API.
- SECURITY REQUIREMENTS
 - The preferred authentication mechanisms are Open ID Connect or Mutual Authentication with X.509 certificates.
 - The rTH POS API supports HTTP Basic Authentication over HTTPS if X.509 mutual authentication is not supported by the requesting the client. For more information about HTTP Basic Authentication, please refer to Common Authorization Headers. The rTH POS API also supports the OAuth 2.0 protocol.

5 INTERFACES

5.1 Common Authorization Headers

Below is the common header required for all the API requests. Please append appropriate values to the authorization header.

Name	Required?	Description
authorization	Yes	Example 1: <code>Authorization: Basic YWRtaW46ZGQwMTgzNTNhOTJkZDBjODU0NDQ3NWJlMDhiYjJlNGZlMTZmYmRhNg==</code> Example 2: <code>Authorization: Bearer 4ae6ce68-4c59-4313-94e2-fcc2932cf5ca</code>

Example:

```
curl -request GET 'https://{host}/rts/public/api/v1/pos/search
header 'Authorization: Basic YWRtaW46ZGQwMTgzNTNhOTJkZDBjODU0NDQ3NWJlMDhiYjJlNGZlMTZmYmRhNg=='
```

5.2 OAuth 2.0 Authorization

The Public API can be configured to accept and authorize requests based on basic authentication or OAuth2.0 standard. Client will be registered in internal authorization server and given client credentials with an endpoint to retrieve authorization tokens from. Below are the steps to retrieve an access token and how to use it to invoke the public API.

Steps to retrieve access token:

1. Invoke the authorization server end point using below request.

Request Type: **POST**

Headers:

Name	Value	Description
Content-Type	x-www-form-urlencoded	The type of content accepted for current request.

Request Body:

Name	Description
client_id	Client id used as a username against the authorization server.
client_secret	Client secret is used as password against the authorization server.
grant_type	Grant type is the type of permissions allowed for the user.

Sample Response (access token length omitted brevity) :

```
{
  "access_token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXLTQwIiwiaWF0IjoiYjI0ZDg5LTQwNTUyOTkxLTRhYTA0GMxYi0yMmMzZTkzZjFjMzk... ",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTQwIiwiaWF0IjoiYjI0ZDg5LTQwNTUyOTkxLTRhYTA0GMxYi0yMmMzZTkzZjFjMzk... ",
  "token_type": "bearer",
  "not-before-policy": 0,
  "session_state": "c3e239e6-7889-4252-9b08-c525bc85370c",
}
```

```

"scope": "email profile"
}

```

2. Retrieve the access token from above request and append it into the Authorization Header as Bearer token.

Example:

Name	Required?	Description
Authorization	Yes	Example 2: <code>Authorization: Bearer {access_token}</code>


5.3 Common HTTP Response Codes

Below are the common HTTP response codes which are returned for each API.

HTTP Response Code	Name	Comment
200	Success	The request was successfully processed by the server.
201	Created	The object requested was created in server.
400	Bad Request	Check that the URL structure and parameters are correct.
401	Unauthorized	Make sure the application has the right permissions and the access token is valid.
404	Resource Not Found	Unable to find the resource in server.
500	Internal server error	Internal server error occurred. Please try again.

6 Status Check API

The Status Check API allows the requester to check the availability and status of the items scanned. Below example request has the required fields used to search for the item.

	rfxcel Traceability Hub (rTH)	rTH-Design-Spec-09
	Design Specification	Page 9 of 17

Requests can be made for a single item or bulk items. The preferred method is to send a request after each item is scanned so that the user is able to determine immediately whether the item should be dispensed or not.

POST “/api/v1/pos/search”

Single Item Request:

This request searches for the availability and status for the item scanned.

```
curl --request POST 'http://{host}/rts/public/api/v1/pos/search'
--header 'Authorization: Basic
YWRtaW46ZGQwMTgzNTNhOTJkZDBjODU0NDQ3NWJlMDhiYjJlNGZlMTZmYmRhNg=='
--header 'Content-Type: application/json'
--data-raw {
{
  "txnId" : "1234",
  "serialNumbers": [
  {
    "productCode": "66666666666675",
    "serialNumber": "12345678901234",
    "lotNumber": "43T216F",
    "itemExpirationDate": "2024-06-30"
  }
  ]
}
```

Bulk Request:

This request searches for the availability and status for the three items that were scanned.

```
curl --request POST 'http://{host}/rts/public/api/v1/pos/search'
--header 'Authorization: Basic
YWRtaW46ZGQwMTgzNTNhOTJkZDBjODU0NDQ3NWJlMDhiYjJlNGZlMTZmYmRhNg=='
--header 'Content-Type: application/json'
--data-raw {
{
  "txnId" : "1234",
  "serialNumbers": [
  {
    "productCode": "66666666666675",
    "serialNumber": "12345678909876",
    "lotNumber": "22F987G",
    "itemExpirationDate": "2024-06-30"
  },
  {
```

```

"productCode": "666666666666675",
"serialNumber": "12345678901234",
"lotNumber": "43T216F",
"itemExpirationDate": "2024-06-30"
},
{
"productCode": "80356845125559",
"serialNumber": "098765432121",
"lotNumber": "4RT4198336",
"itemExpirationDate": "2023-09-01"
}
]
}
}

```

Request Attributes:

txnId (Optional)	Unique transaction IDs generated by the POS system
serialNumbers[]	A list of the items scanned that includes the product code and a serial number for the item.
productNumber	Unique product number of the item scanned
serialNumber	Unique serial number of the item scanned
lotNumber	Lot number of the item extracted from the barcode
itemExpirationDate	Expiration Date of the item extracted from the barcode.


Single Item Response- Successful Response with 201 status code:

Sends the disposition of the item scanned as well as a message indicating whether the item is dispensable or not.

```

{
  "responseStatus": "Success",
  "responseMessage": null,
  "serialNumbers": [
    {

```

	rfxcel Traceability Hub (rTH)	rTH-Design-Spec-09
	Design Specification	Page 11 of 17

```

    "status": "Returned",
    "productCode": "66666666666675",
    "serialNumber": "12345678901234",
    "message": "Item is available to dispense.",
    "statusCode": "dispensable"
  }
],
"txnId": "1234"
}

```

Bulk Response- Successful Response with 201 status code:

Sends the disposition of the items scanned as well as a message indicating whether the item is dispensable or not.

```

{
  "responseStatus": "Success",
  "responseMessage": null,
  "serialNumbers": [
    {
      "status": "Unknown",
      "productCode": "66666666666675",
      "serialNumber": "12345678909876",
      "message": "Unable to find serial number in the system.",
      "statusCode": "dispensable_unknown"
    },
    {
      "status": "Returned",
      "productCode": "66666666666675",
      "serialNumber": "12345678901234",
      "message": "Item is available to dispense.",
      "statusCode": "dispensable"
    },
    {
      "status": "Dispensed",
      "productCode": "80356845125559",
      "serialNumber": "098765432121",
      "message": "Serial number is in Dispensed status. Serial number
should not be
dispensed.",
      "statusCode": "not_dispensable"
    }
  ],
  "txnId": "1234"
}

```

Response Attributes:

responseStatus	The Status Check API will return a success or error response depending on the request sent
responseMessage	Any error event exceptions will be logged in this field.
serialNumbers[]	The list of serial numbers that were scanned along with responses for each
status	The disposition state that the item is currently in
productCode	Product code of the item that was scanned
serialNumber	Serial number of the item that was scanned
message	Message relevant to the item that was scanned. Message suggests whether the item is dispensable or not.
statusCode	Field that suggests whether item is dispensable or not
txnId	Unique ID generated for each request

Status Code Descriptions:

dispensable_unknown	The serial number of the item that was scanned could not be found in the system. This item is still dispensable.
dispensable	This item and the data associated with the item was found in the system and the state transition to dispense is possible. This item is dispensable.
not_dispensable	The serial number was found in the system but was in an invalid disposition to allow the state transition to dispense. This item is not

	dispensable. Invalid dispositions: recalled, destroyed, expired, or dispensed
--	--

List of common errors that may occur for search

```
{
  "timestamp": "2020-11-28T13:24:02.239+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "",
  "path": "/pos/search"
}
```


7 Create Events- JSON Format

Description: The create events REST API will generate GS1 standard events based on EPCIS formatting. Events are generated using POST API requests and the required request body fields vary based on event type you would like to create.

Notes: Event creation can be unsuccessful if input EPCIS event data does not pass rTH event creation validations. For example, attempting to dispense a dispensed serial number will produce an unsuccessful event. As a result, the API returns a **200** HTTP code and the details on why the event did not create successfully.

Event Request Attribute:

txnID	Unique transaction IDs generated by the POS system
eventID	A unique ID created by rTH to identify each event.
eventTimeZoneOffset (optional)	Timezone offset of the location the user is in
EventTime (optional)	Time when the event was created. Defaults to current time if not provided.

	rfxcel Traceability Hub (rTH)	rTH-Design-Spec-09
	Design Specification	Page 14 of 17

serialNumbers	A list of the serial numbers along with additional data that were scanned.
bizLocation	Location of the facility

1. Create Dispense Event

Create an individual dispense event using the format detailed below. The example request has the required fields used to create a dispense event.


POST `"/api/v1/pos/dispense"`

Request:

```
curl --request POST 'http://{host}/rts/public/api/v1/pos/dispense'
--header 'Authorization: Basic
YWRtaW46ZGQwMTgzNTNhOTJkZDBjODU0NDQ3NWJlMDhiYjJlNGZlMTZmYmRhNg=='
--header 'Content-Type: application/json'
--data-raw {
{
  "txnId": "123",
  "eventId": "55589",
  "eventTimeZoneOffset": "+00:00",
  "eventTime": "2022-03-04T08:42:28.003Z",
  "serialNumbers": [
    {
      "productCode": "66666666666675",
      "serialNumber": "12345678901234",
      "lotNumber": "43T216F",
      "itemExpirationDate": "2024-06-30"
    }
    {
      "productCode": "80356845125559",
      "serialNumber": "098765432121",
      "lotNumber": "4RT4198336",
      "itemExpirationDate": "2023-09-01"
    }
  ],
  "bizLocation": "123456"
}
}
```

Successful Response with 201 status code:

A “pending” response status means that the request was received and is queued in the system for further processing. The details and status of the event can be viewed in the rTH UI or queried via the results API.

	rfxcel Traceability Hub (rTH)	rTH-Design-Spec-09
	Design Specification	Page 15 of 17

```
{
  "txId": "123",
  "documentId": "a0616a75-f422-4fb6-b0cd-902080913aef",
  "responseStatus": "Pending",
  "responseMessage": "Events queued for processing"
}
```

Creates a dispense event for the 2 items in the list.

List of common errors that may occur for dispense events.

A “Failed” response status means that the request was not received successfully and that the user would need to try again.

```
{
  "txId": "123",
  "responseStatus": "Failed",
  "responseMessage": "Unable to process request at current time, please contact support."
}
```

2. Create Return Event


Create an individual return event using the format detailed below. The example request has the required fields used to create a return event.

POST “/api/v1/pos/return”

Request:

```
curl --request POST 'http://{host}/rts/public/api/v1/pos/return'
--header 'Authorization: Basic YWRtaW46ZGQwMTgzNTNhOTJkZDBjODU0NDQ3NWJlMDhiYjJlNGZlMTZmYmRhNg=='
--header 'Content-Type: application/json'
--data-raw {
{
  "txnId": "123",
  "15ventide": "55589",
  "eventTimeZoneOffset": "+00:00",
  "eventTime": "2022-03-04T08:42:28.003Z",
  "serialNumbers": [
    {
      "productCode": "66666666666675",
      "serialNumber": "12345678901234",
      "lotNumber": "43T216F",

```

	rfxcel Traceability Hub (rTH)	rTH-Design-Spec-09
	Design Specification	Page 16 of 17

```

    "itemExpirationDate": "2024-06-30"
  },
  {
    "productCode": "80356845125559",
    "serialNumber": "098765432121",
    "lotNumber": "4RT4198336",
    "itemExpirationDate": "2023-09-01"
  }
],
"bizLocation": "123456"
}
}

```

Successful Response with 201 status code:

A “pending” response status means that the request was received and is queued in the system for further processing. The details and status of the event can be viewed in the rTH UI or queried via the results API.

```

{
  "txId": "123",
  "documentId": "89f36bd4-a35e-4a01-b21b-da79d564d0c0",
  "responseStatus": "Pending",
  "responseMessage": "Events queued for processing"
}

```

Creates a Return event for the 2 items in the list.

List of common errors that may occur for dispense events.

A “Failed” response status means that the request was not received successfully and that the user would need to try again.

```

{
  "txId": "123",
  "responseStatus": "Failed",
  "responseMessage": "Unable to process request at current time, please contact support."
}

```

7.1 Get Event Creation Result from asynchronous processing

Get the result for each event created using the create events APIs. The create event response will provide a ‘documentId’ that you can use to get the result.

Request Method:

GET /rts/public/api/v1/events/result/{documentId}

Response Body Fields:

Field	Type	Description
documentId	String	The id for the asynchronous request
message	String	Client friendly string providing the state of the request
status	String	Status of the event request

Examples:

Request: /rts/public/api/v1/events/result/**89f36bd4-a35e-4a01-b21b-da79d564d0c0**

Get result for a document that is not fully processed and is in pending state

```
{
  "documentId": "89f36bd4-a35e-4a01-b21b-da79d564d0c0"
  "message": "Events queued for processing."
  "status": "Pending"
}
```