Responder Enterprise Nurse Call
Real Time Location Systems REST API SDK

Version 0.02

08/03/2018

Version History

Version	Date	Change
0.01	06/28/2018	Initial Version
0.02	08/03/2018	Added RTLS SDK details

Table of Contents

1 Interface Goals

The goal of the API is to allow location and nurse call systems to establish, maintain, and monitor the accuracy of location information flow from the facility's location system to the nurse call system, so that the location information can be utilized within the nurse call applications. The communication must be timely enough to follow staff as they move from room to room without missing a location change.

The API must be flexible to work with the variety of features and capabilities of the majority of nurse call and location suppliers and extensible to grow as market needs dictate additional features.

2 Protocol Considerations

2.1 Differences from Responder 5 API

- Responder Enterprise RTLS uses REST instead of WCF + Raw TCP sockets
- The methods to pull configuration data have been simplified in Responder Enterprise

2.2 RESTful Web Services

Use basic HTTP "Representation state transfer" (aka REST) for all interactions.

2.2.1 Un-acknowledged Event Publishing

All real-time publishing: FROM the Vendor source TO Rauland can be considered one-way and one-shot (at most once message delivery) – badge movement updates are not to be retried and responses from the Rauland system can be ignored for logic processing.

2.2.2 Short Location Change Messages

Location changes, which comprise most of integration traffic, are passed as short messages containing the minimal information required to convey the change. Additional information is not required but is allowed since it may reduce the overall system response time by eliminating the need for clients to look up information. Multiple tag movement messages can be batched into a single REST call, but batching messages is discouraged due to the added latency.

2.2.3 JSON and HTTP messaging

REST uses standard HTTP/S with JSON formatting for all requests and responses

2.3 Reliability

2.3.1 Configuration Queries

The protocol supports queries from the Rauland system to the Vendor source system which allow a for periodic configuration information – list of tag names, list of locations, etc.

2.3.2 Message Lengths

"GET" commands must support REST pagination: The client can request a subset of items, by number, and the server must include a "total number of items" in the response.

2.4 Response Times

Badge movement needs to be communicated near real-time as nurse call location response time expectations are within 15 seconds.

2.5 Version Compatibility

The API conforms to the benefits and limitations of a standard RESTful web service. No versioning system is defined at this time but will be revisited if necessary.

2.6 Custom Protocol Extensions

The API must accommodate differences in features and capabilities of various and location and nurse call Vendor sources. Use of REST allows fields beyond those defined here to be added to any message, with the recipient ignoring the extra fields unless specifically ready to handle them.

2.7 Time Format

All timestamps are formatted per ISO 8601 with combined date and time (including seconds) as Coordinated Universal Time (UTC).
Example: `2018-06-28T13:15:30Z`

3 Security

3.1 Transport

Data transmission can be secured through HTTPS as required.

3.2 Authentication and Authorization

At this time no authentication or authorization mechanism are specified.

4 API Summary

The protocol has two distinct parts:
Realtime: Vendor RTLS source is the client and **Rauland** is the **server**.
Configuration: Vendor RTLS source is the server and **Rauland** is the **client**.

5 API Details

All character strings are limited to 255 characters unless otherwise specified.

Servers are identified by a base URL and commands + parameters are appended to that root path to form a request target.

Example Vendor Server URL = http://vendor.hospital.net/, used below when the vender is the server

Example Rauland Server URL = http://RaulandResponder.hospital.net/api, used below when Rauland is the server

NOTE: in the Rauland Server example above the base URL includes the path /api, which is common ASP MVC controller routing syntax, but is not necessary for this protocol.

5.1 Api Overview

5.1.1 Rauland's server implementation

Method	URL	Payload	Result
GET	/healthcheck		Healthcheck Result
PUT	/movement	Movement Payload	

5.1.2 Vendor RTLS Source's server

Method	URL	Payload	Result
GET	/version		Version Result
GET	/tags?skip={x}&top={y}	Index of first tag and number of tags expected in result	Tags Result
GET	/locations?skip={x}&top={y}	Index of first location and number of locations expected in result	Locations Result

5.2 Setup/Configuration

The **Rauland** system is the **server** and the Vendor source system is a client.

5.2.1 Healthcheck

- **UtcTime** - UTC date and time including seconds formatted per ISO 8601
- **Machine** – computer name of Windows machine that serviced the request
- **Version** – assembly version

5.2.1.1.1 Example Request :

```
PUT http://RaulandResponder.hospital.net/api/healthcheck
```

5.2.1.1.2 Example Result (JSON body)

Healthcheck Result

```
{
    "UctTime": "2018-07-13T12:12:13Z",
    "Machine": "RaulandEnt01.hospital.net",
    "Version":
        {
            "_Major": "2018",
            "_Minor": "7",
            "_Build": "10",
            "_Revision": "1"
        }
}
```

5.3 Realtime Tag Movements

The **Rauland** system is the **server** and the Vendor source system is a client.

5.3.1 Movement

All tag movement update messages are **PUT** and multiple movement message objects can be sent in one message (i.e. an array). The minimal object format parameters are:

- **VendorSourceName** – unique identifier for this RTLS vendor installation and matches the *VendorSourceName* parameter from the *Version* command (below). Note: The enterprise may have many Vendor source installations at one or more hospitals, but all *VendorSourceName* must be unique throughout the entire enterprise.

- **Movements**[]
  - **Tag** – the unique identifier (name) for the tag
  - **Location** – the unique location string (name)
  - **Timestamp** – UTC date and time including seconds

5.3.1.1 Example HTTP

5.3.1.1.1 Example Request :

```
PUT http://RaulandResponder.hospital.net/api/movement
```

5.3.1.1.2 Example Payload (JSON body of PUT)

Movement Payload

```
{
    "VendorSourceName": "RTLSVENDOR1",
    "movements": [
        {
                "tag": "89901",
                "location": "HospitalA-Bldg2-Area1-Room201-Bed2",
                "timestamp": "2018-06-28T13:02:30Z"
        },
        {
                "tag": "98802",
                "location": "HospitalA-AnnexB-Room30-Bed1",
                "timestamp": "2018-06-28T20:25:05Z"
        }
] }
```

5.3.1.2 Example Javascript

NOTE: using JQuery for simplicity

```
var movements = {
        VendorSourceName:"RTLSVENDOR1",
        Movements: [
                {Tag:"89901",Location:"HospitalA-Bldg2-Area1-Room201-Bed2",Timestamp:"2018-06-28T13:02:30Z"},
                {Tag:"98802",Location:"HospitalA-AnnexB-Room30-Bed1",Timestamp:"2018-06-28T20:25:05Z"} ] };
$.ajax({
        url: 'http://RaulandResponder.hospital.net/api/movement',
        type: 'PUT',
        data: movements
});
```

5.4 Configuration Messages

The **RTLS vendor source** system is the **server**, the Rauland system is a client. All information request messages are **GET** methods.

5.4.1 Version

Rauland client requests basic Vendor software information. The minimal response object fields:

- **VendorSourceName** – unique identifier for this RTLS vendor installation and matches the *VendorSourcename* parameter from the *Movement* command above
- **Brand** – manufacturer's name
- **Version** – something useful for debugging
- **Timestamp** – UTC date and time including seconds

5.4.1.1 Example HTTP

5.4.1.1.1 Example Request :

```
GET http://vendor.hospital.net/version
```

5.4.1.1.2 Example Response (JSON body)

Version Result

```
{
    "brand": "Rtls Make",
    "VendorSourceName": "RTLS east wing",
    "version": "Alpha-5.2",
    "timestamp": "2018-06-28T17:48:18Z"
}
```

5.4.2 Tags

Rauland client requests basic Vendor Tag/Badge information, including page ranges (i.e. pagination). The required request parameters:

- **Skip** – the index/identity for the first item to be returned
  - NOTE: if the first requested index is greater than the total amount of tags in the RTLS vendor system, then a response with a correct *Total* amount and empty *tags* is expected
- **Top** – the maximum number of locations to be included in the response
  - NOTE: if the first + count requested is greater than the total amount of tags in the RTLS vendor system, then a response with a correct *Total* is expected with the *tags* array being less than the requested count

The minimal response object fields:

- **Total** – maximum number of tags in the system
- **Tags** – array of strings = the unique names of the tags within the requested range
  - Each tag string can be a maximum of 255 characters

5.4.2.1 Example HTTP

5.4.2.1.1 Example Request :

```
GET http://vendor.hospital.net/tags?skip=50&top=10
```

5.4.2.1.2 Example Response (JSON body)

Tags Result

```
{
    "Total": 8192,
    "tags": [ "badge0198","badge0199","tag001","tag001A","tag001B","tag002","tag003","tag004","tag005","tag006" ]
}
```

5.4.3 Locations

Rauland client requests basic Vendor Location information, including page ranges (i.e. pagination). The required request parameters:

- **Skip** – the index/identity for the first item to be returned
  - NOTE: if the first requested index is greater than the total amount of locations in the RTLS vendor system, then a response with a correct *total* amount and empty *locations* is expected
- **Top** – the maximum number of locations to be included in the response
  - NOTE: if the first + count requested is greater than the total amount of locations in the RTLS vendor system, then a response with a correct *total* is expected with the *locations* array being less than the requested count

The minimal response object fields:

- **Total** – maximum number of tags in the system
- **Locations** – array of strings = the unique names of the location within the requested range
  - Each location string can be a maximum of 255 characters

5.4.3.1 Example HTTP

5.4.3.1.1 Example Request :

```
GET http://vendor.hospital.net/locations?skip=1010&top=10
```

5.4.3.1.2 Example Response (JSON body)

Locations Result


```
{
    "total": 2048,
    "locations": [ "HospitalA-Annex1-Room447","HospitalA-Annex1-Room448","HospitalA-Annex1-Room449-
Bed1","HospitalA-Annex1-Room450-Bed1","HospitalA-Annex1-Room451-Bed2", "HospitalA-Bldg2-Area1-Room201-
Bed2","HospitalA-Bldg2-Area1-Room202-Bed1","HospitalA-Bldg2-Area1-Room202-Bed2","HospitalA-Bldg2-Area1-Room203-
Bed1","HospitalA-Bldg2-Area1-Room204-Bed1" ]
}
```


6 Responder RTLS Integrations SDK Application

6.1 SDK Contents

The Rauland RTLS Integrations SDK consists of:

- This document
- Two (2) OpenAPI/swagger JSON definitions for the Vendor Client and Vendor Server
  - These contracts are suitable for verifying your implementation or using a codegen tool to generate source code templates
  - VendorServer swagger contract.json
  - VendorClient swagger contract.json
- .NET Core Console application that will run a series of tests to validate certain core features that are required to operate with the Rauland Responder Enterprise RTLS Integration
  - This program requires .NET Core 2.1, available from Microsoft

6.2 SDK Test Application Description

The .NET Core Console application (RaulandEnterpriseRtlsSdk.exe) executes a series of tests, mostly http GET (REST) calls, and then opens a Web Service endpoint and listens for *movement* PUT messages. Each test writes a line of text to the console detailing a PASS/FAIL and a brief explanation of the failure texts if applicable. All tests are run (i.e. the program does not stop processing upon the first failure). The Web Service is started after all initial REST call tests are completed. The Web Service will print one message in the Console Window for each *movement* message it receives, including PASS or FAIL messages like the previous tests.

RaulandEnterpriseRtlsSdk.exe requires 2 command-line parameters:

- -u = The URL of the vendor's machine
- -p = the Web Service port number for binding

Example command line:

```
RaulandEnterpriseRtlsSdk.exe -u http://my.server.vendor.net:8080 -p 8081
```

This would start the test with an api call to “http://my.server.vendor.net:8080/tags?skip=0&top=10” for example. After the initial tests complete it would open a Web Service to receive messages like “http://localhost:8081/movements”. The Console will continue running and processing *movement* messages until the user presses the CTRL-C key combination to signal the application to close the web service and exit.

6.3 SDK Test Application Test Details

NOTE: Some of the tests require the test vendor server to have at least 100 tags and 100 locations to thoroughly test some variations of the paging API.

The tests are:

6.3.1 Test Vendor’s Web Service

- Test the 3 API calls which should be implemented by the RTLS Vendor Source
  - API: Version
    - `VendorVersionApiTest`: check response contains all 4 properties as NOT null/empty
    - `VendorVersionApiTimestampTest`: return timestamp is correct (within 5 minutes of "now"...in UTC)
  - API: Tags
    - `VendorTagsApiTotalCountTest`: return value has a TOTAL
    - `VendorTagsApiShouldReturnRequestedNumberOfTagsTest`: single tag query returns = top (i.e. if I do a query for skip=0&top=10 then I get exactly 10 tags for my query)
    - `VendorTagsApiSkipReturnsCorrectSubsetTest`: confirm SKIP works correctly
      - 2 requests: first get tags skip=0&top=10
      - 2 requests: first get tags skip=5&top=10, make sure that the first 5 tags in query 2 matches the last 5 in query 1
    - `VendorTagsApiShouldReturnCorrectTotalTest`: download entire list of tags from vendor, confirm Tag count matches TOTAL
    - `VendorTagsApiTagsShouldBeUniqueTest`: download entire list of tags from vendor, confirm Tags are ALL UNIQUE
  - API: Locations
    - `VendorLocationsApiShouldReturnLocationsTest`: Vendor system has non-empty locations response
    - `VendorLocationsApiTotalShouldBeAtLeast100Test`: Vendor system contains at least 100 locations
    - `VendorLocationsApiShouldReturnRequestedNumberOfLocationsTest`: single location query returns = top (i.e. if I do a query for skip=0&top=10 then I get exactly 10 locations for my query)
    - `VendorLocationsApiSkipReturnsCorrectSubsetTest`: confirm SKIP works correctly
      - 2 requests: first get lcations skip=0&top=10

        - 2 requests: first get locations skip=5&top=10, make sure that the first 5 locations in query 2 matches the last 5 in query 1
    - `VendorLocationsApiShouldReturnCorrectTotalTest`: download entire list of locations from vendor, confirm Location count matches TOTAL
    - `VendorLocationsApiLocationsShouldBeUniqueTest`: download entire list of locations from vendor, confirm Locations are ALL UNIQUE

6.3.2 Test Vendor's client calls

- Test the API calls expected from the vendor
    - API: Movement
        - `RequestBodyTest`: confirm the JSON body includes all required parameters
        - `RequestTimestampTest`: timestamp parameters are correct (within 5 minutes of "now"...in UTC)
        - `ValidateVendorSourceNameMatchTest`: confirm the VendorSourceName parameter matches the VendorSourceName used in the prior client tests
        - `ValidateVendorTagsExistenceTest`: confirm the tag(s) in the movement message exist in tag queries from prior tests
        - `ValidateVendorLocationsExistenceTest`: confirm the location(s) in the movement message exist in location queries from prior tests