SunGuide®:

SPARR
Interface Control Document

SunGuide-SPARR-ICD-8.2

Prepared for:

Florida Department of Transportation
Traffic Engineering and Operations
Office 605 Suwannee Street, M.S. 90

Tallahassee, Florida 32399-0450

(850) 410-5600

October 5, 2022

Document Control Panel			
File Name:	SunGuide-SPARR-ICD-6.0.1.doc		
File Location:	SunGuide CM Repository		
	Name	**Initial**	**Date**
Created By:	Roger Strain, SwRI	RLS	11/28/11
Reviewed By:	Tucker Brown, SwRI	TJB	02/11/11
	Tucker Brown, SwRI	TJB	01/09/14
	AJ Skillern	AJS	10/05/22
Modified By:	Roger Strain, SwRI	RLS	06/22/12
	Tucker Brown, SwRI	TJB	02/02/16
	Tucker Brown, SwRI	TJB	10/05/22
Completed By:			

Table of Contents

List of Figures

List of Acronyms

ATMS	Advanced Traffic Management System
AVL	Automatic Vehicle Location
DOT	Department of Transportation
FDOT	Florida Department of Transportation
GZIP	GNU Zip
HTTP	Hypertext Transport Protocol
HTTPS	Hypertext Transport Protocol (Secured)
IM	Incident Management
ITS	Intelligent Transportation Systems
ITN	Invitation to Negotiate
JSON	Javascript Object Notification
RR	Road Rangers
SPARR	Smartphone Application for Road Rangers
SwRI	Southwest Research Institute
TMC	Traffic Management Center
XML	Extensible Markup Language

REVISION HISTORY

Revision	Date	Changes
5.0.5-Draft	January 28, 2011	Initial Release
6.2	February 2, 2016	Updated for 6.2 Release

1. Scope

1.1 Document Identification

This Interface Control Document (ICD) describes the interface between individual Smartphone Application for Road Rangers (SPARR) smartphone clients and the SPARR Automatic Vehicle Location / Road Ranger (AVL/RR). This interface does not use the standard XML communications described in the general ICD. This ICD defines web service methods and objects which are used in communication between the phone application and SunGuide driver.

1.2 Project Overview

The Florida Department of Transportation (FDOT) SunGuide Support, Maintenance and Development Contract addresses the necessity of supporting, maintaining and performing enhancement development efforts to the SunGuide software. The SunGuide software is a set of Intelligent Transportation System (ITS) software that allows the control of roadway devices as well as information exchange across a variety of transportation agencies and is deployed throughout the state of Florida. The SunGuide software is based on ITS software available from the state of Texas; with significant customization and development of new software modules to meet the needs of the FDOT. The following figure provides a graphical view of the software to be developed.

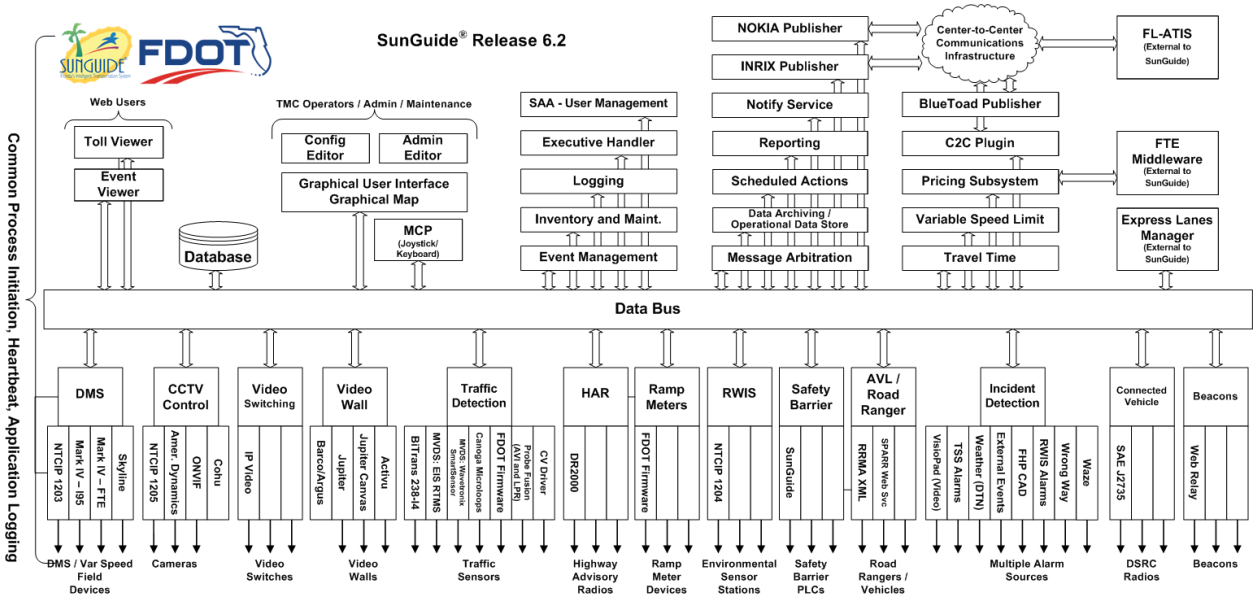

Figure 1.1 - High-Level Architectural Concept

1.3 Related Documents

The following documents were used to develop this document:

- FDOT Scope of Services: *BDQ69, Standard Written Agreement for SunGuide Software Support, Maintenance, and Development, Exhibit A: Scope of Services.* July 1, 2010.
- Notice to Proceed: Letter to SwRI for BDQ69, July 1, 2010
- Letter of Authorization 004: Letter to SwRI for BDQ69, October 20, 2010.
- SunGuide Project website: http://sunguidesoftware.com.

1.4 Contacts

The following are contact persons for the SunGuide software project:

- Fred Heery, ITS Section, Traffic Engineering and Operations Office Central Office, fred.heery@dot.state.fl.us, 850-410-5606
- Derek Vollmer, ITS Section, Traffic Engineering and Operations Office Central Office, Derek.Vollmer@dot.state.fl.us, 850-410-5615
- Clay Packard, Atkins Project Manager, clay.packard@dot.state.fl.us, 850-410-5623
- David Chang, Atkins Project Advisor, david.chang@dot.state.fl.us, 850-410-5622
- Tucker Brown, SwRI Project Manager, tbrown@swri.com, 210-522-3035
- Roger Strain, SwRI Software Project Manager, rstrain@swri.org, 210-522-6295

2. Web Service

A web service interface is exposed for use by the SPARR application. Requests are made via HTTP/S requests which are handled by an HTTP server hosted inside the driver process. The config.xml file specifies the name of the web service in the webServiceName tag, and the port in the webServicePort tag. If the useSSL configuration tag is set to true, the application will attempt to use HTTPS instead of HTTP; this requires appropriate configuration on the host to enable SSL support for the specified port. Information on configuring a server for SSL can be found in the SunGuide 6.0 VDD.

Once running, HTTP or HTTPS requests should be made using a URL in the following format:
`http(s)://[hostname]:[webServicePort]/[webServiceName]/[methodName]?[params]`

Any parameters to the requested method should be appended as standard HTTP GET parameters (param1=value1&param2=value2). DateTime parameters should be formatted as "MM/dd/yyyy hh:mm:ss a", where the final “a” is either AM or PM. Requests must provide credentials of an account on the host machine via standard HTTP authentication.

Responses to all requests are GZIPped prior to sending to reduce transmission overhead. Clients may have to manually unzip the responses if their interface does not do so automatically. Most responses will be in the form of JSON strings which follow the structure of the various objects marked as return types. Some methods may return data in a different format; see each method's documentation.

Many requests return a **SparrResult** object; this object includes both a success/failure indication of the request's result, as well as other information about the current session. Clients should examine these result objects to ensure their session is still recognized as valid, and to determine whether they need to request updates to any events, vehicles, or databases.

When initially starting, only a subset of calls may be issued prior to starting a full session. The calls which may be made any time include CheckForUpdates, GetAllVehicles, GetAvlDb, GetConfiguration, GetEmDb, and GetVehicle. When ready to begin a patrolling session, StartSession should be called. After this, all methods will be available, assuming any prerequisite conditions are met.

2.1 Web Method Calls

This section defines each method call available through the web service.

2.1.1 AddActivities

SparrResult AddActivities (string *clientId*, DateTime *dataTimestamp*, DateTime *currentTimestamp*, string *eventId*, string *activities*)

Adds an activity to a specified event for the current vehicle.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.
eventId	Identifier of the event arrived at.

activities	The string representation of a JSON array consisting of one or more activities. An activity consists of its name and the quantity for the activity. An example of a single activity: {"activity":"Abandoned","quantity":0} An example of multiple activities: [{"activity":"Abandoned","quantity":0},{"activity":"Running","quantity":6},{"activity":"Assist FHP","quantity":0},{"activity":"Cold Patch Asphalt","quantity":9}]

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.2 AddActivity

`SparrResult AddActivity (string clientId, DateTime dataTimestamp, DateTime currentTimestamp, string eventId, string activity, int quantity)`

Adds an activity to a specified event for the current vehicle.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.
eventId	Identifier of the event arrived at.
activity	Activity to be added, short name
quantity	Number of activity items provided/used.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.3 AddEventComment

`SparrResult AddEventComment (string clientId, DateTime dataTimestamp, DateTime currentTimestamp, string eventId, string commentType, string commentText)`

Adds a comment to an event record.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.
eventId	Identifier of the event to add the comment to.
commentType	Type of comment to add.

commentText	Text of comment to add.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.4 AddInvolvedVehicle

```
SparrResult AddInvolvedVehicle (string clientId, DateTime dataTimestamp, DateTime currentTimestamp, string eventId, string tagNumber, string make, string model, string color, string state)
```

Adds an involved vehicle to the specified event.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.
eventId	Identifier of the event arrived at.
tagNumber	Tag number of the vehicle
make	Make of the vehicle
model	Model of the Vehicle
color	Color of the Vehicle
state	State of the license plate

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.5 ArriveAtEvent

```
SparrResult ArriveAtEvent (string clientId, DateTime dataTimestamp, DateTime currentTimestamp, string eventId)
```

Indicates that a SPARR client has arrived at an event.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.
eventId	Identifier of the event arrived at.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.6 ReportBulkAvlStatus

```
SparrResult ReportBulkAvlStatus(Stream gzipedBulkStatus)
```

Provides a method of inserting a large number of vehicle location history records to the database without flooding the operator map with the traffic.

Parameters:

gzippedBulkStatus	Wrapped "reportAvlDataRequest". Package includes clientId node, dateTime node, and reportAvlDataRequest node.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.7 CheckForUpdates

```
SparrResult CheckForUpdates (string clientId)
```

Requests a list of pending updates which should be requested by the client.

Parameters:

clientId	Unique client identifier.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.8 CloseEvent

```
SparrResult CloseEvent (string clientId, DateTime dataTimestamp, DateTime currentTimestamp, string eventId)
```

Indicates a SPARR client wishes to close an event.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.
eventId	Id of the event to attempt to close.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.9 CreateEvent

```
SparrResult CreateEvent (string clientId, DateTime dataTimestamp, DateTime currentTimestamp, string eventType, int locationId)
```

Indicates a SPARR client wishes to generate a new event.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.

eventType	Type of event to create.
locationId	Location at which the event should be created.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.10 DepartFromEvent

```
SparrResult DepartFromEvent (string clientId, DateTime dataTimestamp, DateTime currentTimestamp, string eventId)
```

Indicates that a SPARR client has departed from an event. This request may fail if the client has not performed any activities at that event.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.
eventId	Identifier of the event departed from.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.11 EndSession

```
SparrResult EndSession (string clientId, string mileage)
```

Indicates a SPARR client has logged out from a patrol session.

Parameters:

clientId	Unique client identifier.
mileage	Vehicle mileage at the end of the session. There is a system parameter that determines whether this is required or optional. If optional, this may be omitted.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.12 GetAllEvents

```
List<SparrEvent> GetAllEvents (string clientId)
```

Requests a list of all events currently known to the system, with their current status.

Parameters:

clientId	Unique client identifier.

Returns:

List of all current events with status.

2.1.13 GetAllVehicleEvents

`List<SparrEvent> GetAllEvents (string clientId)`

Requests a list of all events involving road rangers currently known to the system.

Parameters:

clientId	Unique client identifier.

Returns:

List of all current events with road rangers assigned.

2.1.14 GetAllVehicles

`List<SparrAvlrrVehicle> GetAllVehicles (string clientId)`

Requests a list of all vehicles currently known to the system, with their current status.

Parameters:

clientId	Unique client identifier.

Returns:

List of all current vehicles with status.

2.1.15 GetAvlDb

`Stream GetAvlDb (string clientId, string dbId)`

Retrieves the current SQLite3 database containing AVL configuration information. This data is returned as a binary stream, rather than a JSON string. The data is still GZIPped prior to transmission.

Parameters:

clientId	Unique client identifier.
dbId	Hashed identifier of the database to retrieve.

Returns:

Binary data file content.

2.1.16 GetConfiguration

`SparrConfiguration GetConfiguration ()`

Retrieves SPARR client configuration parameters.

Returns:

Configuration information for the client.

2.1.17 GetEmDb

`Stream GetEmDb (string clientId, string dbId)`

Retrieves the current SQLite3 database containing EM configuration information. This data is returned as a binary stream, rather than a JSON string. The data is still GZIPped prior to transmission.

Parameters:

clientId	Unique client identifier.
dbId	Hashed identifier of the database to retrieve.

Returns:

Binary data file content.

2.1.18 GetEvent

```
SparrEvent GetEvent (string clientId, string eventId)
```

Requests the current status of a specific event.

Parameters:

clientId	Unique client identifier.
eventId	Identifier of the event to retrieve data for.

Returns:

Status of requested event. If the event has been deleted, null should be returned.

2.1.19 GetVehicle

```
SparrAvlrrVehicle GetVehicle (string clientId, string vehicleId)
```

Requests the current status of a specific vehicle.

Parameters:

clientId	Unique client identifier.
vehicleId	Identifier of the vehicle to retrieve data for.

Returns:

Status of requested vehicle. If the vehicle has been deleted, null should be returned.

2.1.20 ModifyPatrol

```
SparrResult ModifyPatrol (string clientId, string radio, string beat)
```

Indicates a SPARR client has modified parameters regarding a patrol session.

Parameters:

clientId	Unique client identifier.
radio	Radio the operator has specified. (Blank if not available.)
beat	Beat the operator has specified. (Blank if not available.)

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.21 ReportAvailabilityStatus

```
SparrResult ReportAvailabilityStatus (string clientId, string newStatus)
```

Reports an availability status change from a SPARR client. This request may fail for a variety of reasons, depending on the current state of the vehicle and the state requested.

Parameters:

clientId	Unique client identifier.
newStatus	Newly selected status.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.22 ReportAvlStatus

`SparrResult ReportAvlStatus (string clientId, DateTime dataTimestamp, DateTime currentTimestamp, int latitude, int longitude, int altitude, int speed, int heading)`

Allows a vehicle to report its current location information.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.
latitude	Current latitude of the device in microseconds. (Integer.MinValue (-2147483648) if not available.)
longitude	Current longitude of the device in microseconds. (Integer.MinValue (-2147483648) if not available.)
altitude	Current altitude in feet. (Integer.MinValue (-2147483648) if not available.)
speed	Current speed in MPH. (Integer.MinValue (-2147483648) if not available.)
heading	Curernt heading in degrees, with north as zero. (Integer.MinValue (-2147483648) if not available.)

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.23 SetEventToUnresolved

`SparrResult SetEventToUnresolved (string clientId, DateTime dataTimestamp, DateTime, currentTimestamp, string eventId)`

Indicates a SPARR client wishes to set an event status to “Unresolved”.

Parameters:

clientId	Unique client identifier.
dataTimestamp	Device timestamp when the data was collected.
currentTimestamp	Device timestamp when the data was sent.

eventId	Id of the event to attempt to close.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.24 StartSession

```
SparrResult StartSession (string clientId, string operatorId, string vehicleId, string radio, string beat, bool overrideVehicleCheck, bool isNewSession, string mileage)
```

Indicates a SPARR client has logged in and is ready to begin a patrol session.

Parameters:

clientId	Unique client identifier.
operatorId	Identifier of the operator who has logged in to the SPARR session.
vehicleId	Vehicle the operator has specified.
radio	Radio the operator has specified. (Blank if not available.)
beat	Beat the operator has specified. (Blank if not available.)
overrideVehicleCheck	If true, override the check for another operator already being assigned to the vehicle.
isNewSession	If true, treat this as a new session regardless of previous state.
mileage	Vehicle mileage at the start of the session. There is a system parameter that determines whether this is required or optional. If optional, this may be omitted.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.1.25 UpdateClientId

```
SparrResult UpdateClientId (string oldClientId, string newClientId)
```

Indicates that the identifier associated to a client has been updated.

Parameters:

oldClientId	Previously assigned client identifier.
newClientId	Newly assigned client identifier.

Returns:

Response indicating success or failure, and pending data to be retrieved.

2.2 Web Method Objects

The following sections document the various objects which may be sent in response to web method requests.

2.2.1 SparrActivity

Represents a SPARR view of an EM activity.

- string **Name**
  Name of the Activity.
- int **Id**
  Id of the Activity.
- int **Quantity**
  Quantity of the Activity.
- bool **Quantifiable**
  Whether the activity accepts a quantity.

2.2.2 SparrAvlrrVehicle

Represents subset of data regarding a vehicle reported to SPARR.

- override string **ClientId**
  Client ID associated to the vehicle, if any.
- **SparrId Id**
  Identifier of the vehicle.
- **SparrId Operator**
  Identifier of the driver of this vehicle, if set.
- string **VehicleName**
  Display name of this vehicle, if set.
- string **OpStatus**
  Current op status of the vehicle.
- string **Beat**
  Beat the vehicle is assigned to patrol.
- string **Radio**
  Radio assigned to the vehicle.
- string **Telephone**
  Phone assigned to the vehicle.
- **SparrId EventId**
  Event the vehicle is arrived at.
- string **AvailabilityStatus**
  Current availability status of the vehicle.

2.2.3 SparrConfiguration

Configuration information sent to a SPARR client on startup.

- List< **SparrPhone** > **PhoneNumbers**
  List of phone numbers to display as part of a quick dial interface.
- int **UpdateCheckFrequency**
  How many seconds the client should wait between checking for updated data.

- int **PositionUpdateFrequency**
  How many seconds the client should wait between sending position update messages.
- int **DbUpdateDelay**

How many seconds the client should wait before requesting a database update following the detection of a new hash value.

2.2.4 SparrEvent

Represents subset of data regarding an event presented to SPARR.

- **SparrId Id**
  Identifier of the event.
- string **Status**
  Current state of the event.
- string **Type**
  Current type of the event.
- int **LocationId**
  Identifier of the location of this event.
- string **LocationDesc**
  Parsed description of the location.
- string **BlockageDesc**
  Parsed description of the location.
- List< **SparrEventLaneBlockage** > **LaneBlockages**
  Collection of lanes at the event location, including blockage information.
- List< **SparrEventResponder** > **Responders**
  Collection of event vehicle responder records.
- List< **SparrEventAgencyResponder** > **AgencyResponders**
  Collection of event agency responder records.
- List< **SparrInvolvedVehicle** > **InvolvedVehicles**
  Collection of event involved vehicle records.

2.2.5 SparrEventAgencyResponder

Represents data reported for a specific event agency responder.

- string **Name**
  Name of the responder vehicle.
- string **Status**
  Status of the vehicle with respect to the event.

2.2.6 SparrEventLaneBlockage

Represents subset of data regarding blocked lanes reported to SPARR.

- int **LaneIndex**
  Order of the lane, starting at the left.

- int **LaneTypeId**
  Type of lane.
- string **LaneTypeName**
  Lane Type Classification.
- string **LaneStatus**
  Current state of the lane (blocked, clear, etc.)

2.2.7 SparrEventResponder

Represents data reported for a specific event responder.

- string **Name**
  Name of the responder vehicle.
- string **Status**
  Status of the vehicle with respect to the event.
- List< **SparrActivity** > **Activities**
  Number of activities performed by this responder.

2.2.8 SparrId

Wraps a SunGuide ID for use in the SPARR interface.

- string **Id**
  Gets or sets the id.
- string **CenterId**
  Gets or sets the center id.
- string **ResourceType**
  Gets or sets the type of the resource.
- string **ProviderName**
  Gets or sets the name of the provider.

2.2.9 SparrInvolvedVehicle

Represents data about a vehicle involved in an event which is sent to a SPARR client.

- int **Id**
  Id of the Vehicle.
- string **Make**
  Make of the Vehicle.
- string **Model**
  Model of the Vehicle.
- string **Color**
  Color of the Vehicle.
- string **Tag**
  Vehicle Tag Number.
- string **State**

State of the Vehicle.

2.2.10 SparrPhone

A phone number record displayed to a SPARR client.

- string **Name**
  Name of the owner of the phone number.
- string **Number**
  The number to dial. (Should be in a format which can be auto-dialed by an application.)

2.2.11 SparrResult

Specifies result of a SPARR client request, including information about pending data retrieval required.

- bool **Success**
  Whether the request was successfully forwarded from the driver.
- bool **SessionValid**
  Whether the current SPARR session remains valid. (This may be false due to a duplicate login, restart of the driver, or other reason.)
- string **ErrorMessage**
  If a failure occurred, the error that should be displayed to the user.
- string **AvlDbHash**
  MD5 hash of the currently available AVL configuration database.
- string **EmDbHash**
  MD5 hash of the currently available EM configuration database.
- **SparrId LastCreatedEventId**
  ID of the last event created by this road ranger.
- List< **SparrId** > **UpdatedVehicles**
  List of vehicles which have updates which have not yet been requested by this client.
- List< **SparrId** > **UpdatedEvents**
  List of events which have updates which have not yet been requested by this client.

2.3 AVL Configuration Database

The following sections document the various tables present in the AVL configuration database generated by the driver. This database is an SQLite database file. Note that SQLite does not support a native Boolean type; any fields below defined as Boolean will contain a string value where "Y" indicates true and "N" indicates false. Each table and view also includes a primary key column named _id, which contains an auto-incrementing integer identifier.

2.3.1 AvailabilityStatus Table

This table contains all defined availability status values reported by the AVL/RR subsystem.

- integer **Id**
  The database id of the availability status.

- text **Name**
  The name of the availability status.
- Boolean **InService**
  Whether the status is one that indicates the Road Ranger is currently in service.
- Boolean **Assisting**
  Whether the status is one that indicates the Road Ranger is currently providing assistance at an event.
- Boolean **CanBeDispatched**
  Whether the status is one that indicates the Road Ranger is currently valid to be dispatched to an event.
- Boolean **DefaultStartShift**
  Whether the status is what the system should assign when a Road Ranger begins a shift.
- Boolean **DefaultEndShift**
  Whether the status is what the system should assign when a Road Ranger ends a shift.
- Boolean **DefaultAssisting**
  Whether the status is what the system should assign when a Road Ranger arrives at an event.
- Boolean **DefaultPatrolling**
  Whether the status is what the system should assign when a Road Ranger departs an event.

2.3.2 Beat Table

This table contains all defined beats (routes) reported by the AVL/RR subsystem.

- integer **Id**
  The database id of the beat.
- text **Name**
  The name of the beat.

2.3.3 Operator Table

This table contains all defined Road Ranger operators (drivers) reported by the AVL/RR subsystem.

- integer **Id**
  The database id of the operator.
- text **OperatorName**
  The identifier of the operator.
- text **FirstName**
  The first name of the operator.
- text **LastName**
  The last name of the operator.
- text **Password**
  The plain text password the operator must use to authenticate with the system.

2.3.4 Radio Table

This table contains all defined radios reported by the AVL/RR subsystem.

- integer **Id**
  The identifier of the radio.
- text **Name**
  The identifier of the radio

2.3.5 OperatorNameView View

This view provides a more friendly list of available operator identifiers.

- text **DisplayName**
  The identifier and full name of an operator.

2.4 EM Configuration Database

The following sections document the various tables present in the EM configuration database generated by the driver. This database is an SQLite database file. Note that SQLite does not support a native Boolean type; any fields below defined as Boolean will contain a string value where "Y" indicates true and "N" indicates false. Each table and view also includes a primary key column named _id, which contains an auto-incrementing integer identifier.

2.4.1 Activities Table

This table contains all defined event responder activity types reported by the EM subsystem.

- text **Name**
  The name of the availability status.
- integer **SortOrder**
  A number indicating the order in which the item should appear in a list.
- Boolean **Quantifiable**
  The name of the availability status.

2.4.2 CommentTypes Table

This table contains all defined event comment types reported by the EM subsystem.

- text **Name**
  The name of the comment type.
- integer **SortOrder**
  A number indicating the order in which the item should appear in a list.

2.4.3 EventType Table

This table contains all defined event types reported by the EM subsystem.

- text **Name**
  The name of the event type.
- integer **SortOrder**
  A number indicating the order in which the item should appear in a list.

2.4.4 Location Table

This table contains all defined locations reported by the EM subsystem.

- integer **Id**
  A unique identifier of the location.
- integer **ReferencePointId**
  The unique identifier of the reference point used by this location.
- text **Description**
  A textual description of the location suitable to display to a user.
- integer **RoadwayId**
  The unique identifier of the roadway used by this location.
- text **Direction**
  The name of the direction used by this location.
- numeric **Latitude**
  The latitude of the location in microdegrees.
- numeric **Longitude**
  The longitude of the location in microdegrees.
- float **Distance**
  The distance of the location from the current GPS position of the device. (This value should be automatically updated by the client if and when necessary. Initial values from the SPARR driver will not be useful.)

2.4.5 ReferencePoint Table

This table contains all defined location reference points reported by the EM subsystem.

- integer **Id**
  A unique identifier of the reference point.
- text **Name**
  The name of the reference point.
- integer **SortOrder**
  A number indicating the order in which the item should appear in a list.

2.4.6 Roadway Table

This table contains all defined roadways reported by the EM subsystem.

- integer **Id**
  A unique identifier of the roadway.
- text **Name**
  The name of the roadway.
- integer **SortOrder**
  A number indicating the order in which the item should appear in a list.

2.4.7 RoadwayDirection Table

This table contains all defined roadway and direction pairs reported by the EM subsystem. Roadways only travel in directions for which they have a matching pair in this table.

- integer **RoadwayId**
  The unique identifier of the roadway for this data pair.
- text **Direction**
  The name of the direction for this data pair.

2.4.8 States Table

This table contains all defined vehicle tag states reported by the EM subsystem.

- text **Name**
  The name of the state.
- integer **SortOrder**
  A number indicating the order in which the item should appear in a list.

2.4.9 VehicleColor Table

This table contains all defined vehicle colors reported by the EM subsystem.

- text **Name**
  The name of the color.
- integer **SortOrder**
  A number indicating the order in which the item should appear in a list.

2.4.10 VehicleMake Table

This table contains all defined vehicle makes reported by the EM subsystem.

- integer **Id**
  A unique identifier of the vehicle make.
- text **Name**
  The name of the vehicle make.
- integer **SortOrder**
  A number indicating the order in which the item should appear in a list.

2.4.11 VehicleModel Table

This table contains all defined vehicle models reported by the EM subsystem.

- integer **MakeId**
  The unique identifier of the model this make is associated with.
- text **Name**
  The name of the vehicle make.
- integer **SortOrder**
  A number indicating the order in which the item should appear in a list.

3. Notes

Information about HTTP can be found at the World Wide Web Consortium (W3) website at http://www.w3.org. Information about JSON can be found at http://www.json.org/.