### Document Control Panel

<table>
<thead>
<tr>
<th>File Name:</th>
<th>SunGuide-DB-SUB-ICD-6.2.doc</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Location:</td>
<td>SunGuide CM Repository</td>
</tr>
<tr>
<td>CDRL:</td>
<td>2-7.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Initial</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created By:</td>
<td>Meredith Moczygemma, SwRI</td>
<td>MRM</td>
</tr>
<tr>
<td>Reviewed By:</td>
<td>Steve Dellenback, SwRI</td>
<td>SWD</td>
</tr>
<tr>
<td></td>
<td>Steve Novosad, SwRI</td>
<td>SEN</td>
</tr>
<tr>
<td>Modified By:</td>
<td>Tucker Brown, SwRI</td>
<td>TJB</td>
</tr>
<tr>
<td>Completed By:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table of Contents

1. Scope ................................................................................................................................. 1
   1.1 Document Identification ............................................................................................ 1
   1.2 Project Overview ........................................................................................................ 1
   1.3 Related Documents ..................................................................................................... 2
   1.4 Contacts ...................................................................................................................... 3

2. Data ...................................................................................................................................... 4
   2.1 Initial Client Communication ..................................................................................... 4
       2.1.1 Authenticate ........................................................................................................ 4
       2.1.2 Retrieve Data ....................................................................................................... 7
   2.2 Status Update Models ................................................................................................. 18
       2.2.1 Generic Update Response and Message Models ................................................. 18
       2.2.2 Add and Modify Update Pattern Models ............................................................. 21
       2.2.3 Delete Update Pattern Model ............................................................................. 24
   2.3 Client Connection Updates ......................................................................................... 25
   2.4Subsystem Commands ................................................................................................. 25
List of Figures

Figure 1-1 - High-Level Architectural Concept ........................................................................... 2
Figure 2-1 – authenticateReq ........................................................................................................ 5
Figure 2-2 - Required username Element ..................................................................................... 5
Figure 2-3 - Required password Element ..................................................................................... 6
Figure 2-4 – authenticateResp ........................................................................................................ 6
Figure 2-5 – authenticateData ......................................................................................................... 7
Figure 2-6 – retrieveDataReq Model ............................................................................................ 9
Figure 2-7 - statusList element required for the Data Bus ............................................................... 9
Figure 2-8 - retrieveDataResp Model .......................................................................................... 10
Figure 2-9 – retrieveData .............................................................................................................. 11
Figure 2-10 – statusList Model .................................................................................................... 11
Figure 2-11 - resourceType Model ............................................................................................... 12
Figure 2-12 - id Model .................................................................................................................. 13
Figure 2-13 - Sample Configuration File Setup ........................................................................... 14
Figure 2-14 - subscribeReq Model ............................................................................................... 15
Figure 2-15 - Subscription Data Type Model ............................................................................... 15
Figure 2-16 - subscribeResp Model ............................................................................................ 16
Figure 2-17 - subscribeData Model .............................................................................................. 17
Figure 2-18 - Subscription Data Type Model ............................................................................... 18
Figure 2-19 - Generic Update Message Model ........................................................................... 20
Figure 2-20 - Generic Update Response Model ........................................................................... 21
Figure 2-21 – genericUpdateData Model ..................................................................................... 21
Figure 2-22 - Add Update Response Pattern Model ...................................................................... 22
Figure 2-23 – addUpdateData Model .......................................................................................... 23
Figure 2-24 - resourceType Model .............................................................................................. 24
Figure 2-25 - Delete Update Response Pattern Model ................................................................. 25
Figure 2-26 - deleteUpdateData Model ....................................................................................... 25
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATMS</td>
<td>Advanced Traffic Management System</td>
</tr>
<tr>
<td>DOT</td>
<td>Department of Transportation</td>
</tr>
<tr>
<td>FDOT</td>
<td>Florida Department of Transportation</td>
</tr>
<tr>
<td>IM</td>
<td>Incident Management</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transportation Systems</td>
</tr>
<tr>
<td>ITN</td>
<td>Invitation to Negotiate</td>
</tr>
<tr>
<td>SwRI</td>
<td>Southwest Research Institute</td>
</tr>
<tr>
<td>TMC</td>
<td>Traffic Management Center</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
## REVISION HISTORY

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0</td>
<td>November 4, 2002</td>
<td>Initial Release</td>
</tr>
<tr>
<td>6.2</td>
<td>February 3, 2016</td>
<td>6.2 Release Update</td>
</tr>
</tbody>
</table>

1. Scope

1.1 Document Identification

This Interface Control Document (ICD) describes the system interface between the individual SunGuide data provider subsystems and the Data Bus. It is necessary for each of the providers to uniformly communicate with the Data Bus to ensure schema compatibility. This ICD template will also allow the Data Bus to easily expand to support new subsystems without modifying the SunGuide system. Refer to the SunGuide-General-ICD-1.0.0 document for details regarding data transfer.

1.2 Project Overview

The Florida Department of Transportation (FDOT) is conducting a program that is developing 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. The goal of the SunGuide℠ software is to have a common software base that can be deployed throughout the state of Florida. The SunGuide℠ software development effort is based on ITS software available from both the states of Texas and Maryland; significant customization of the software is being performed as well as the development of new software modules. The following figure provides a graphical view of the software to be developed:
1.3 Related Documents

The following documents were used to develop this document:

- Notice to Proceed: Letter to Southwest Research Institute® (SwRI®) for BDQ69, July 1, 2010.

1.4 Contacts

The following are contact persons for the SunGuideSM 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. Data

The following sections detail the XML transactions that can be exchanged between client and server applications.

2.1 Initial Client Communication

This section explains initial communication between the Data Bus and the data provider subsystems. The Data Bus Client Interface Manager (CIM) ICD may be referenced for more information regarding Data Bus user client communication with the Data Bus.

2.1.1 Authenticate

Before any other commands can be sent, the Data Bus must send an authenticateReq to register with the provider subsystem. The authenticate request is a transaction type which contains two additional fields: username and password. The username is the name of the application or user (e.g. databus) that is connecting to the provider subsystem. The Data Bus must have an associated password that the provider can retrieve from the database. The password sent as part of this request is encrypted using Message Digest 5 (MD5) hashing.

The provider system uses the username and password to verify the client’s privileges. If the authentication is successful, a securityToken will be returned to the Data Bus. If not successful, an error message will be returned. The securityToken returned by the provider to the Data Bus must be sent with each additional request and will be used to validate the Data Bus’s ability to perform the request.

Each provider subsystem must have a username and password for the Data Bus with the following minimum/maximum permissions: system access, retrieve resource data, and subscribe to resource data and status changes. The username and hashed password value used by the Data Bus are specified in the configuration file.
The first XML request sent by the client should be this authenticating request.

```xml
<xs:element name="authenticateReq">
    <xs:annotation>
        <xs:documentation>
The first XML request sent by the client should be this authenticating request.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="TransactionType">
                <xs:sequence>
                    <xs:element name="username" type="identifier">
                        <xs:annotation>
                            <xs:documentation>An identifier representing the client application.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="password" type="xs:string">
                        <xs:annotation>
                            <xs:documentation>The password corresponding to this request.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
```

**Figure 2-1 – authenticateReq**

<table>
<thead>
<tr>
<th>diagram</th>
<th>username</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An identifier representing the client application.</td>
</tr>
<tr>
<td>type</td>
<td>identifier</td>
</tr>
<tr>
<td>facets</td>
<td>minLength 1</td>
</tr>
<tr>
<td>annotation</td>
<td>documentation</td>
</tr>
</tbody>
</table>

**Figure 2-2 - Required username Element**

<table>
<thead>
<tr>
<th>diagram</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The password corresponding to this request.</td>
</tr>
<tr>
<td>type</td>
<td>Restriction of xs:hexBinary</td>
</tr>
<tr>
<td>facets</td>
<td>minLength 6</td>
</tr>
<tr>
<td>annotation</td>
<td>documentation</td>
</tr>
</tbody>
</table>

source

```xml
<xs:element name="username" type="identifier">
    <xs:annotation>
        <xs:documentation>An identifier representing the client application.</xs:documentation>
    </xs:annotation>
</xs:element>
```
The response for an authenticate request includes the security token. The security token is then sent with each additional request from the Data Bus.

**Figure 2-3 - Required password Element**

```xml
<xs:annotation>
  <xs:element name="authenticateResp">
    <xs:annotation>
      <xs:documentation>Response received for an authenticate request</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="ResponseType"/>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
</xs:annotation>
```

**Figure 2-4 – authenticateResp**
2.1.2 Retrieve Data

When a security token is received from a provider subsystem, the Data Bus sends a `retrieveDataReq` to the provider. Each data provider is required to model its `retrieveDataReq` and `retrieveDataResp` after the example request and response in Figure 2-6 and Figure 2-8, respectively. When retrieving data from a provider, the Data Bus selects only the `statusList` element.
A request sent by client to retrieve data from the system

**diagram**

**type** extension of RequestType

**children** refId icdVersion username securityToken statusList

**attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerName</td>
<td>identifier</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>providerType</td>
<td>identifier</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>documentation</td>
<td>type</td>
<td></td>
<td>true</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**source**

```xml
<xs:element name="retrieveDataReq">
  <xs:annotation>
    <xs:documentation>A request sent by client to retrieve data from the system</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="RequestType">
        <xs:sequence>
          <xs:element name="statusList" type="xs:boolean" default="false" minOccurs="0">
            <xs:annotation>
              <xs:documentation>True if the client requests to retrieve all resource status information from the data provider.</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:any>
            <xs:annotation>
              <xs:documentation>Any other subsystem-specific data types can be included in this request; the Data Bus only requests the statusList element above.</xs:documentation>
            </xs:annotation>
          </xs:any>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

SunGuide-DB-SUB-ICD-6.2
The statusList element returned in the retrieveDataResp stores an unbounded set of resource elements, each containing the required id and status elements for the requested data. These resource elements should be named with the appropriate resource type name (e.g. resourceType should be replaced with dms, camera, monitor, etc.). The id element uniquely defines a resource item in the Data Bus status DOM (Document Object Model) trees by including mandatory attributes for the provider name, resource type, and center Id. Moreover, all status update response and message schemas used by the data providers must also use this id element to protect the integrity of status data stored by the Data Bus. Subsystem-specific status data is stored and wrapped by the status element.
Figure 2-8 - retrieveDataResp Model
Figure 2-9 – retrieveData

Figure 2-10 – statusList Model
The resource instance contains status data.

**Figure 2-11 - resourceType Model**
Following successful authentication and status DOM tree initialization, the Data Bus detects and handles status updates from the providers by subscribing to each element within the `subscriptions` element in the configuration file for the pertinent provider, using a similar subscribe schema to the model depicted in Figure 2-14. The `typeAStatus`, `typeBStatus`, and `typeCStatus` elements would be replaced with the appropriate subsystem-specific subscription types (i.e. cameraStatus, monitorData, etc.).

In order for the Data Bus to ensure accurately stored status data at all times, each provider subsystem must make certain that the subscriptions specified in the configuration file encompass all possible status data changes for the retrieved subsystem status information. Each data provider must also populate the `statusUpdates` element in the configuration file (Figure 2-13) for the respective subsystem, imbedding all resource type (i.e. camera, monitor, dms, etc.) elements within this tag. Each resource type element within the `statusUpdates` tag maps the appropriate status update message and response names (i.e. updateConnectionMsg, changeCameraStateResp, etc.) for the subscribed XML updates. Each of these update tags may include the optional `action` attribute, specifying whether the update follows either the `add`, `modify`, or `delete` schema patterns addressed in Error! Reference source not found. (Error! Reference source not found.).
Figure 2-13 - Sample Configuration File Setup

A request sent by client to subscribe to data updates from the system

- refId: The reference id is a unique identifier assigned by the client.
- icdVersion: The version of the icd being used.
- username: The user who sent this request.
- securityToken: This token is provided to the client upon authorization.

- typeAStatus: Request updates to status of all typeA resources within the data provider subsystem.
- typeBStatus: Request updates to status of all typeB resources within the data provider subsystem.
- typeCStatus: Request updates to status of all typeC resources within the data provider subsystem.

Type extension of RequestType

children refId icdVersion username securityToken typeAStatus typeBStatus typeCStatus
The `subscribeResp` (Figure 2-16) will return true values for data to which the Data Bus has successfully subscribed. False values will be returned for data types where the subscription failed. Each data type requested for subscription by the Data Bus must be successful or
subscription fails, and the provider’s status DOM tree is cleared. Upon successful subscription, if any changes occur to the data to which the Data Bus is subscribed, the Data Bus will receive unsolicited responses with updated data. When subscribed XML responses and messages are sent to the Data Bus from the providers, the pertinent status DOM tree stored in the Data Bus is modified accordingly. All messages and responses in the configuration file must set the providerName attribute in the root element. These subscribed responses and messages must also extend the id element in Figure 2-12.

![Image of subscribeResp model]

**Figure 2-16 - subscribeResp Model**
The response data for a subscribe request

```
<xs:complexType name="subscribeData">
  <xs:annotation>
    <xs:documentation>The response data for a subscribe request</xs:documentation>
  </xs:annotation>
  <xs:complexContent extension="responseData">
    <xs:sequence>
      <xs:element name="typeAStatus" type="xs:boolean" default="false" minOccurs="0">Request updates to status of all typeA resources within the data provider subsystem. (Set to true if the subscription was made successfully).
      </xs:element>
      <xs:element name="typeBStatus" type="xs:boolean" default="false" minOccurs="0">Request updates to status of all typeB resources within the data provider subsystem. (Set to true if the subscription was made successfully).
      </xs:element>
      <xs:element name="typeCStatus" type="xs:boolean" default="false" minOccurs="0">Request updates to status of all typeC resources within the data provider subsystem. (Set to true if the subscription was made successfully).
      </xs:element>
    </xs:sequence>
  </xs:complexContent>
</xs:complexType>
```

Figure 2-17 - subscribeData Model
2.2 Status Update Models

All XML messages and responses sent to the Data Bus from a provider subsystem require the securityToken element, indicating the desired client recipient, and the root-level providerName attribute. (Refer to the SunGuide General ICD document for more information on XML base response and message data formats). Moreover, the Data Bus requires all provider subsystems to incorporate the following status update message and response models in subsystem-specific schemas.

2.2.1 Generic Update Response and Message Models

The following two sample schemas indicate the general status update models accepted by the Data Bus. The message model is constructed so that the first customized message element, the id element, is followed by all pertinent update status tags. Each status tag following the id element should be tag names that are either the resource item’s status tag itself, or one of its sub tags, as specified in the statusList element of the provider’s retrieveDataResp description. (Thus, in the figures below, updateTagA and updateTagB are elements found within the status element of this sample subsystem’s retrieveDataResp.)

The Data Bus searches breadthfirst for an id element in the XML update message. Once this element is found, all elements listed in the same level of the XML document as the id element are considered to be status update tags that the Data Bus will attempt to replace in the pertinent status DOM tree.

The response model is similar to the message model, except that the id and update status tags are enclosed by the ResponseType’s data element.

For the case that there are two identically named elements within a resource item’s status element, the topmost element closest to the root level of the XML document will be replaced within the Data Bus status DOM tree. And if the Data Bus is unable to find the specified update tag for the respective resource item within the status DOM tree, a replacement will not be made for that update element.
If the `parentId` attribute is set for this model, the Data Bus will attempt update the DOM tree specified by the `resourceType` by first searching for the `parentId` in the tree, and then if found, the specified identifier (text stored in `id` element). If this secondary `id` element is found in the DOM tree under the located `parentId`, elements at the same level as the found `id` will be searched and updated as appropriate with the specified status updates.

```xml
<xs:complexType name="genericUpdateMsg">
  <xs:annotation>
    <xs:documentation>This is a sample message used to update a status resource item.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="MessageType">
      <xs:sequence>
        <xs:element ref="id"/>
        <xs:element ref="updateTagA"/>
        <xs:element ref="updateTagB"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
<xs:documentation>The string identifying monitor to display the camera on.</xs:documentation>
</xs:element>
</xs:element>
<xs:element name="updateTagA" type="identifier">
<xs:annotation>
<xs:documentation>The updated element, TagA, to be replaced in the Data Bus's status DOM tree for this status resource item.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="updateTagB">
<xs:annotation>
<xs:documentation>The updated element, TagB, to be replaced in the Data Bus's status DOM tree for this status resource item.</xs:documentation>
</xs:annotation>
</xs:element>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>

Figure 2-19 - Generic Update Message Model

diagram

ResponseType

-refld

The reference id is a unique identifier assigned by the client.

-icdVersion

The version of the icd being used.

-securityToken

The response can either have an error or any other data.

error

The error contains an integer code as well as the text of the error message.

data

Response data to be set by individual XML responses.

<xs:element name="genericUpdateResp">
<xs:annotation>
<xs:documentation>This is a sample response used to update a status resource item.</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:complexContent>
<xs:extension base="ResponseType"/>
</xs:complexContent>
</xs:complexType>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerName</td>
<td>identifier</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>providerType</td>
<td>identifier</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

attributes

annotation

source
2.2.2 Add and Modify Update Pattern Models

When a provider adds a new resource item to the system, an update response similar to the add template pattern in Figure 2-22 should be utilized and captured in the list of status updates in the configuration file. Once again, the resource elements should be named with the appropriate resource type name (e.g. resourceType should be replaced with dms, camera, monitor, etc.).

Figure 2-20 - Generic Update Response Model

Figure 2-21 – genericUpdateData Model
update tag name specified in the configuration file for this update response should include the optional action attribute, specifying that the update follows an add schema pattern (Figure 2-13).

The optional modify update pattern is identical to the addUpdateResp below. The notable difference for usage of this pattern is that the update tag in the configuration file for this particular status modifier should set the action attribute to modify, instead of add to trigger suitable Data Bus handling behavior (Figure 2-13). Furthermore, the data providers are not required to implement the modify pattern for their individual systems, as the generic update message and response models above will suffice for this effort. This design pattern offers only another option to help ease customization of each system to comply with the Data Bus.

![Diagram of the ResponseType schema](image)

**Figure 2-22 - Add Update Response Pattern Model**
The data returned when a new status resource item is added to the system.

Diagram:

```
addUpdateData → resourceType
```

Type: extension of responseData

Children: resourceType

Annotation: The data returned when a new status resource item is added to the system.

Source:

```
<xs:complexType name="addUpdateData">
  <xs:annotation>
    <xs:documentation>The data returned when a new status resource item is added to the system.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="responseData">
      <xs:sequence>
        <xs:element name="resourceType" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>The resource instance contains status data.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="id"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="status">
          <xs:annotation>
            <xs:documentation>Stores the status information for the resource item. (This status element should match the same status element as indicated in the statusList element of the retrieveDataResp for this provider subsystem.)</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="id"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 2-23 – addUpdateData Model
2.2.3 Delete Update Pattern Model

When a provider removes a resource item from the system, an update response similar to the delete template pattern in Figure 2-25 should be utilized and captured in the list of status updates in the configuration file. The update tag name specified in the configuration file for this update response should include the optional action attribute, specifying that the update follows a delete schema pattern (Figure 2-13).
Figure 2-25 - Delete Update Response Pattern Model

Figure 2-26 - deleteUpdateData Model

2.3 Client Connection Updates

While the client is connected, a clientDisconnectMsg will be sent when a Data Bus client disconnects from the Data Bus. This update message will contain the pertinent security token and username of the disconnected Data Bus client. (Refer to the SunGuide General ICD for more information regarding this message).

2.4 Subsystem Commands

All provider requests, messages, and responses must extend TransactionType and use the appropriate Req, Msg, and Resp suffixed nomenclature to be successfully routed from client to subsystem and back. In addition, each transaction sent from a client to a subsystem must contain the optional attribute, providerName, to be routed. This providerName must match the provider specification defined in the Data Bus configuration file or an error is returned.

The provider subsystem ICDs may be referenced for more information regarding command and control requests, responses, and messages routed by the Data Bus.