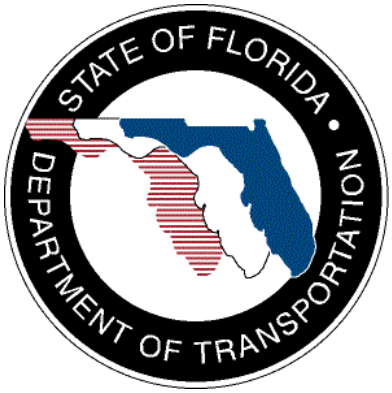
SunGuide[SM]:

Data Bus Subsystem
Provider Template
Interface Control Document

SunGuide-DB-SUB-ICD-1.0.0



Prepared for:

Florida Department of Transportation
Traffic Engineering and Operations Office
605 Suwannee Street, M.S. 90
Tallahassee, Florida 32399-0450
(850) 410-5600

November 14, 2004

Document Control Panel			
File Name:	SunGuide-DB-SUB-ICD-1.0.0.doc		
File Location:	SunGuide CM Repository		
CDRL:	2-7.1		
Name		**Initial**	**Date**
Created By:	Meredith Moczygemba, SwRI	MRM	11/14/04
Reviewed By:	Steve Dellenback, SwRI	SWD	11/14/04
	Steve Novosad, SwRI	SEN	11/14/04
Modified By:			
Completed By:			

Table of Contents

List of Figures

List of Acronyms

ATMS	Advanced Traffic Management System
DOT	Department of Transportation
FDOT	Florida Department of Transportation
IM	Incident Management
ITS	Intelligent Transportation Systems
ITN	Invitation to Negotiate
SwRI	Southwest Research Institute
TMC	Traffic Management Center
XML	Extensible Markup Language

REVISION HISTORY

Revision	Date	Changes
1.0.0	November 4, 2002	Initial Release

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[SM] software. The SunGuide[SM] 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[SM] software is to have a common software base that can be deployed throughout the state of Florida. The SunGuide[SM] 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:

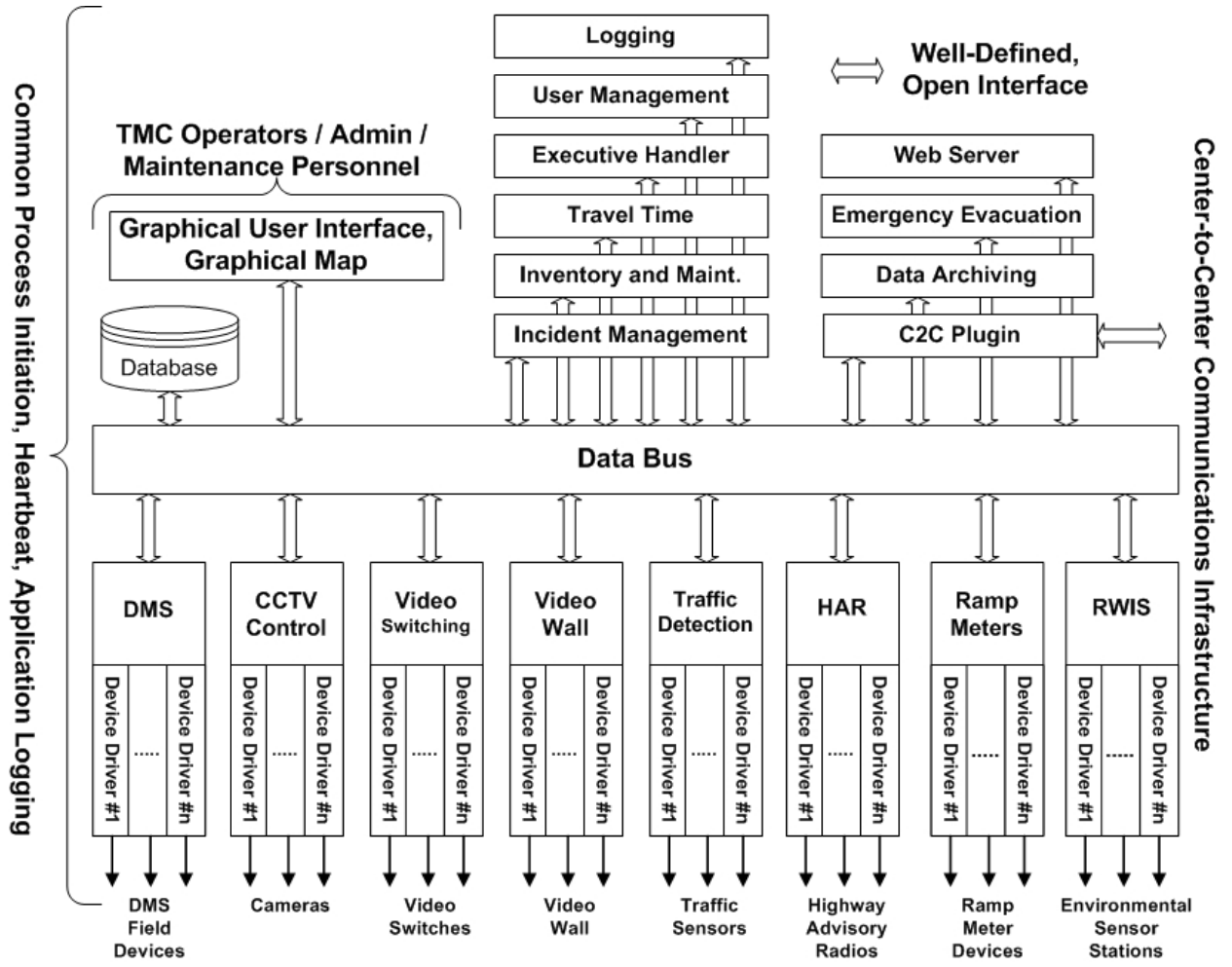


Figure 1-1 - High-Level Architectural Concept

The SunGuide[SM] development effort spans approximately two years. After the development, the software will be deployed to a number of Districts and Expressway Authorities throughout Florida and support activities will be performed.

1.3 Related Documents

The following documents were used to develop this document:

- SwRI Qualification Response: *Response to the Invitation to Negotiate (ITN): Statewide Transportation Management Center Software Library System, Negotiation Number: ITN-DOT-02/03-9025-RR*, SwRI Proposal No. 10-35924, dated: November 18, 2002.
- SwRI Technical Proposal: *Technical Proposal for Invitation to Negotiate (ITN): Statewide Transportation Management Center Software Library System, Negotiation Number: ITN-DOT-02/03-9025-RR*, SwRI Proposal No. 10-35924, dated: January 31, 2003.

- SwRI Cost Proposal: *Cost Proposal for Invitation to Negotiate (ITN): Statewide Transportation Management Center Software Library System, Negotiation Number: ITN-DOT-02/03-9025-RR,* SwRI Proposal No. 10-35924, dated: January 31, 2003.
- SwRI BAFO letter: *Southwest Research Institute® Proposal No. 10-35924, "Invitation to Negotiate (ITN): Statewide Transportation Management Center Software Library System", Reference: Negotiation Number: ITN-DOT-02/03-9025-RR*, dated: May 5, 2003.
- FDOT procurement document: *Invitation To Negotiate (ITN), Negotiation Number: ITN-DOT-02/03-9025-RR, Statewide Transportation Management Center Software Library System*, dated: October 21, 2002.
- FDOT Scope of Services: *Statewide Transportation Management Center Software Library System: Scope of Services*, September 22, 2003.
- FDOT Requirements Document: *Statewide Transportation Management Center Software Library System: Requirements Specification*, June 3, 2003.
- Southwest Research Institute, *TMC Software Study*, November 15, 2001.
- Southwest Research Institute, *Introduction to an Operational Concept For the Florida Statewide Library*, FDOT – OCD – 1.0, March 31, 2002.
- World Wide Web Consortium (W3) website: http://www.w3.org.
- SunGuide℠ Project website: http://sunguide.datasys.swri.edu.

1.4 Contacts

The following are contact persons for the SunGuide℠ software project:

- Elizabeth Birriel, ITS Central Office, elizabeth.birriel@dot.state.fl.us, 850-410-5606
- Liang Hsia, FDOT Project Manager, liang.hsia@dot.state.fl.us, 850-410-5615
- John Bonds, Senior ITS Specialist, jbonds@pbsj.com, 408-873-2514
- David Chang, ITS Specialist, David.Chang@dot.state.fl.us, 850-410-5622
- Steve Dellenback, SwRI Project Manager, sdellenback@swri.org, 210-522-3914
- Robert Heller, SwRI Software Project Manager, rheller@swri.org, 210-522-3824
- Charlie Wallace, PBF Deputy Project Manager, WallaceC@pbworld.com, 352-374-6635
- John Schumitz, PBF Software Project Manager, schumitz@pbworld.com, 301-816-1852

The following are contacts that will be used by the SunGuide℠ software project team to assure consistency with other FDOT projects and FDOT procedures:

- Dan Baxter, PB Farradyne, FDOT C2C Project, baxter@pbworld.com, 407-587-7809
- David Lambert, University of North Florida, RWIS, jlambert@unf.edu, 904-620-3881
- Bob Colins, PBS&J, Emergency Evacuation, bobcolins@pbsj.com, 850-575-1800
- John Fain, FDOT, Comptroller, john.fain@dot.state.fl.us, 850-921-7332
- Leslie Jacobson, PB Farradyne, Ramp Metering, jacobsonl@pbworld.com, 206-382-5290

2. Data

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

2.1 Initial Client Communication

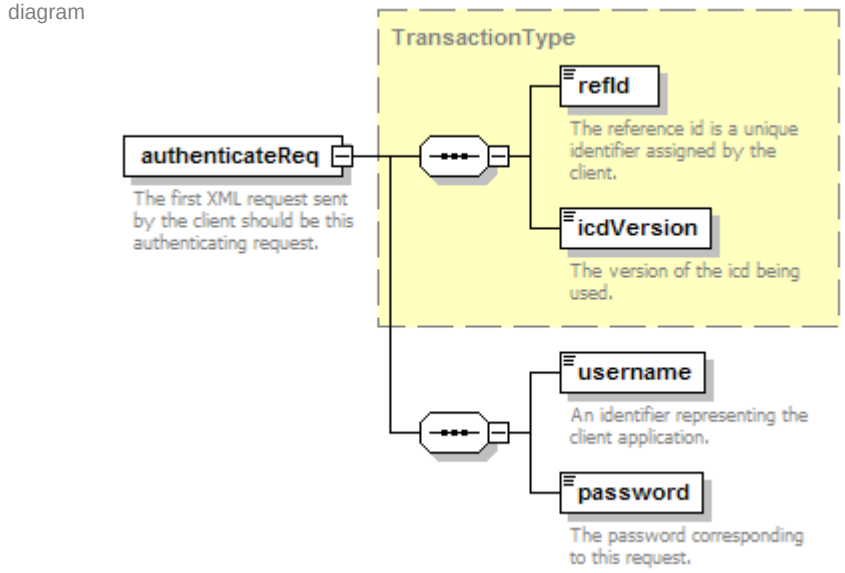
This section explains initial communication between the Data Bus and the data provider subsystems. The Data Bus Client Interface Manager (CIM) ICD may be referenced for more information regarding Data Bus user client communication with the Data Bus.

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 mininum/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.



type extension of **TransactionType**

children **refId** **icdVersion** **username** **password**

attributes	Name	Type	Use	Default	Fixed	Annotation
	providerName	identifier	optional			
	providerType	identifier	optional			
annotation	documentation	The first XML request sent by the client should be this authenticating request.				

source

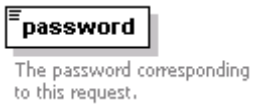
```
<xs:element name="authenticateReq">
 <xs:annotation>
  <xs:documentation>The first XML request sent by the client should be this authenticating request.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="TransactionType">
    <xs:sequence>
     <xs:element 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

diagram	
type	**identifier**
facets	minLength 1 maxLength 30 whiteSpace preserve
annotation	documentation An identifier representing the client application.

source	`<xs:element name="username" type="identifier">`
	`<xs:annotation>`
	`<xs:documentation>An identifier representing the client application.</xs:documentation>`
	`</xs:annotation>`
	`</xs:element>`

Figure 2-2 - Required username Element

diagram	
type	Restriction of **xs:hexBinary**
facets	minLength 6 maxLength 30
annotation	documentation The password corresponding to this request.

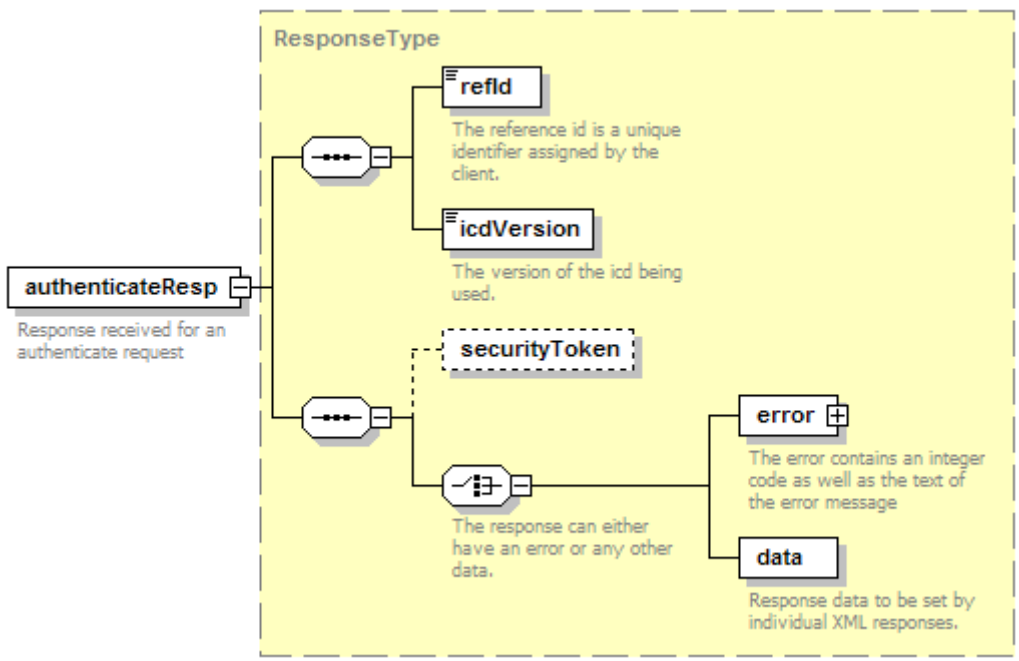
source	`<xs:element name="password">`
	`<xs:annotation>`
	`<xs:documentation>The password corresponding to this request.</xs:documentation>`

```
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:hexBinary">
      <xs:maxLength value="30"/>
      <xs:minLength value="6"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Figure 2-3 - Required password 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.

diagram



type Extension of **ResponseType**

children **refId icdVersion securityToken error data**

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	optional			
providerType	identifier	optional			

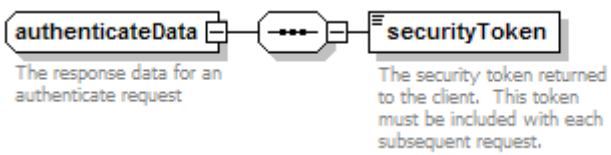
annotation
documentation Response received for an authenticate request

source

```
<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>
```

Figure 2-4 – authenticateResp

diagram



type Extension of **responseData**

children **securityToken**

annotation documentation The response data for an authenticate request

source

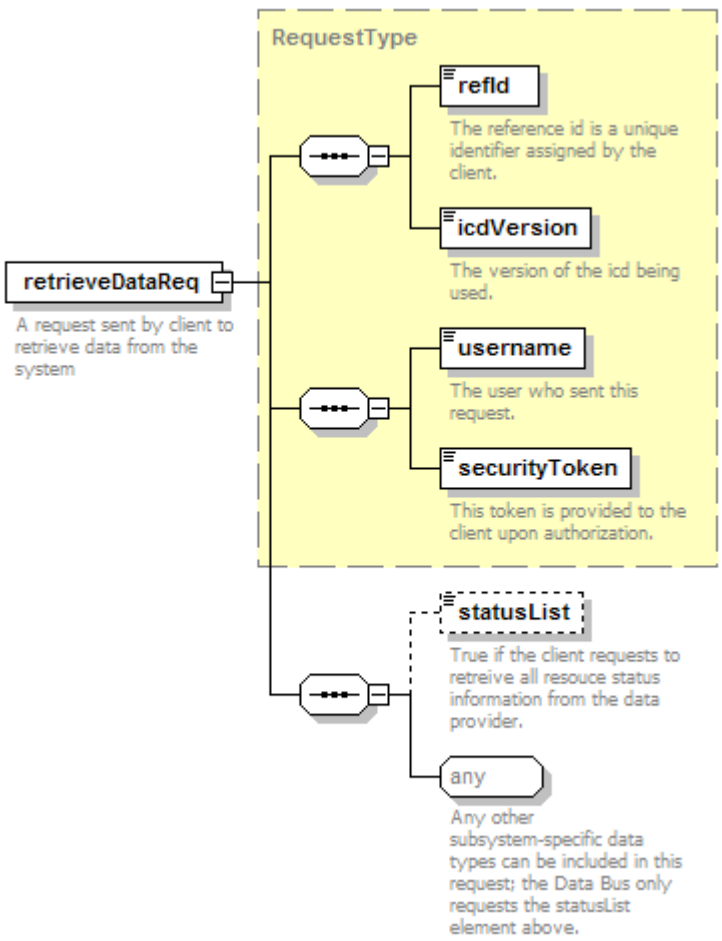
```
<xs:complexType name="authenticateData" abstract="0">
 <xs:annotation>
  <xs:documentation>The response data for an authenticate request</xs:documentation>
 </xs:annotation>
 <xs:complexContent>
  <xs:extension base="responseData">
   <xs:sequence>
    <xs:element name="securityToken">
     <xs:annotation>
      <xs:documentation>The security token returned to the client.  This token must be included with each subsequent request.</xs:documentation>
     </xs:annotation>
     <xs:simpleType>
      <xs:restriction base="xs:string">
       <xs:minLength value="6"/>
       <xs:maxLength value="30"/>
      </xs:restriction>
     </xs:simpleType>
    </xs:element>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Figure 2-5 – authenticateData

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.

diagram



type extension of **RequestType**

children **refId** **icdVersion** **username** **securityToken** **statusList**

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	optional			
providerType	identifier	optional			

annotation documentation A request sent by client to retrieve data from the system

source

```
<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 retreive all resouce 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>
```

Figure 2-6 – retrieveDataReq Model

diagram

type **xs:boolean**

annotation documentation True if the client requests to retreive all resouce status information from the data provider.

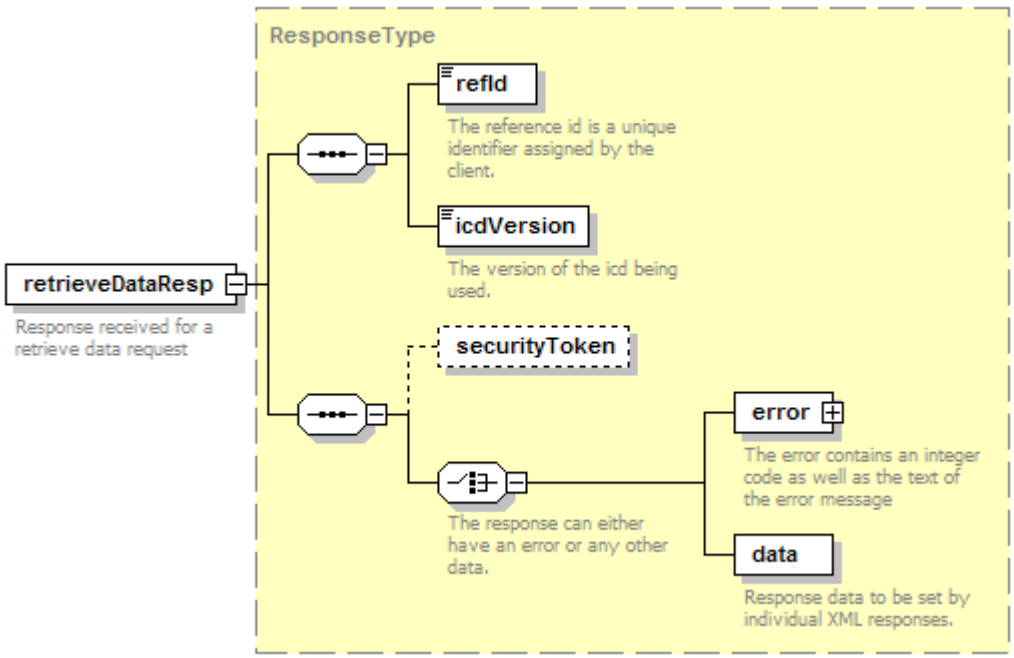
source

```
<xs:element name="statusList" type="xs:boolean" default="false" minOccurs="0">
  <xs:annotation>
    <xs:documentation>True if the client requests to retreive all resouce status information from the data provider.</xs:documentation>
  </xs:annotation>
</xs:element>
```

Figure 2-7 - statusList element required for the Data Bus

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

diagram



type extension of **ResponseType**

children **refId** **icdVersion** **securityToken** **error** **data**

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	optional			
providerType	identifier	optional			

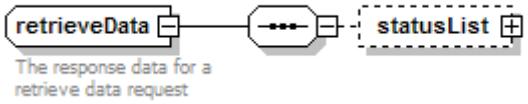
annotation documentation Response received for a retrieve data request

source

```
<xs:element name="retrieveDataResp">
  <xs:annotation>
    <xs:documentation>Response received for a retrieve data request</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="ResponseType"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

Figure 2-8 - retrieveDataResp Model

diagram



type extension of **responseData**

children **statusList**

annotation documentation The response data for a retrieve data request

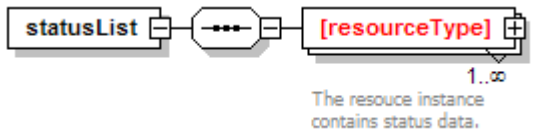
source

```
<xs:complexType name="retrieveData" abstract="0">
  <xs:annotation>
    <xs:documentation>The response data for a retrieve data request</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="responseData">
      <xs:sequence>
```

```
      <xs:element name="statusList" minOccurs="0">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="resourceType" maxOccurs="unbounded">
          <xs:annotation>
           <xs:documentation>The resouce instance contains status data.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
           <xs:sequence>
            <xs:element ref="id"/>
            <xs:element name="status">
             <xs:annotation>
              <xs:documentation>Stores the status information for the resource item.</xs:documentation>
             </xs:annotation>
            </xs:element>
           </xs:sequence>
          </xs:complexType>
         </xs:element>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
```

Figure 2-9 – retrieveData

diagram



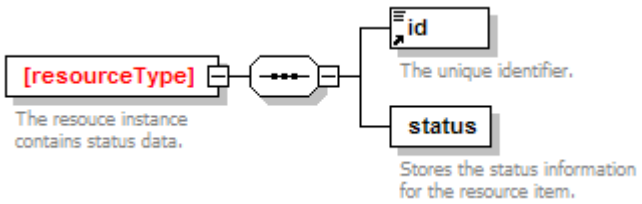
children **[resourceType]**

source

```
<xs:element name="statusList" minOccurs="0">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="[resourceType]" maxOccurs="unbounded">
    <xs:annotation>
     <xs:documentation>The resouce instance contains status data.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
     <xs:sequence>
      <xs:element ref="id"/>
      <xs:element name="status">
       <xs:annotation>
        <xs:documentation>Stores the status information for the resource item.</xs:documentation>
       </xs:annotation>
      </xs:element>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Figure 2-10 – statusList Model

diagram



children **id** **status**

annotation documentation The resouce instance contains status data.

source

```
<xs:element name="[resourceType]" maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>The resouce instance contains status data.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="id"/>
   <xs:element name="status">
    <xs:annotation>
     <xs:documentation>Stores the status information for the resource item.</xs:documentation>
    </xs:annotation>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Figure 2-11 - resourceType Model

diagram

type extension of **identifier**

facets

minLength	1
maxLength	30
whiteSpace	preserve

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	required			
resourceType	identifier	required			
centerId	identifier	required			
parentId	identifier	optional			

annotation documentation The unique identifier.

source

```
<xs:element name="id">
 <xs:annotation>
  <xs:documentation>The unique identifier.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="identifier">
    <xs:attribute name="providerName" type="identifier" use="required">
     <xs:annotation>
      <xs:documentation source="The name of the provider or subsystem to which this item belongs"/>
     </xs:annotation>
    </xs:attribute>
    <xs:attribute name="resourceType" type="identifier" use="required">
     <xs:annotation>
      <xs:documentation source="The type of resource of this item (e.g., link, dms, camera)"/>
     </xs:annotation>
    </xs:attribute>
    <xs:attribute name="centerId" type="identifier" use="required">
     <xs:annotation>
```

```
        <xs:documentation source="The name of the center that owns this item"/>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="parentId" type="identifier" use="optional">
      <xs:annotation>
        <xs:documentation source="The parent id name, if one exists"/>
      </xs:annotation>
    </xs:attribute>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
</xs:element>
```

Figure 2-12 - id 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.).

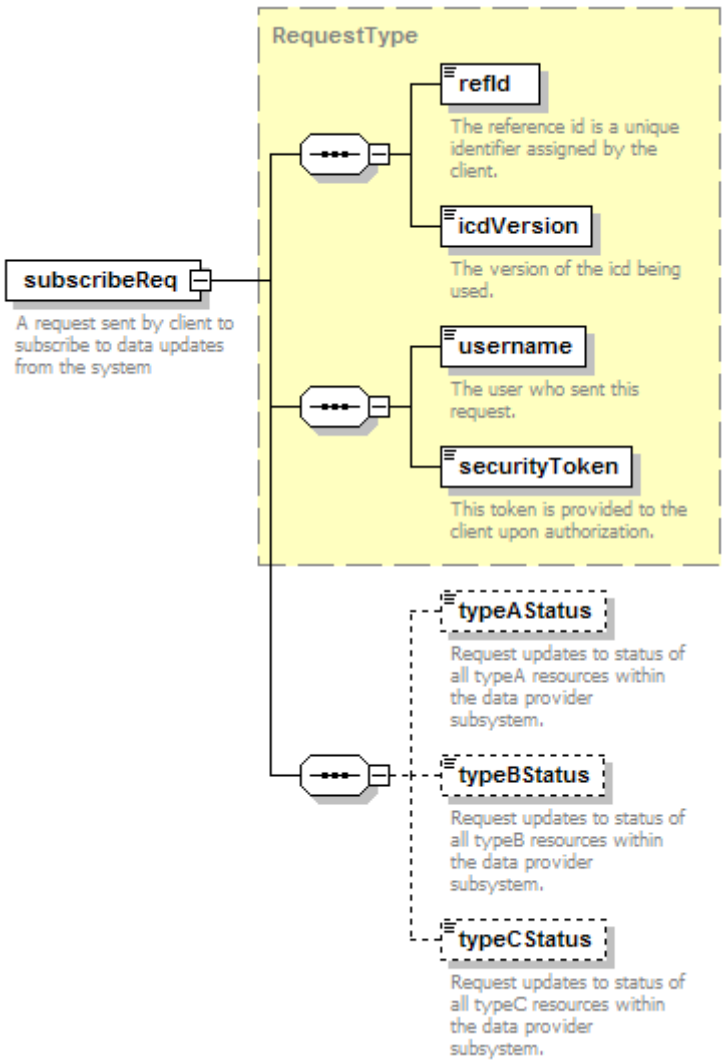
In order for the Data Bus to ensure accurately stored status data at all times, each provider subsystem must make certain that the subscriptions specified in the configuration file encompass all possible status data changes for the retrieved subsystem status information. Each data provider must also 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.**).

```
<dataProviderName>
    .
    .
  <subscriptions>
      <subscriptionType1Name/>
      <subscriptionType2Name/>
  </subscriptions>
  <statusUpdates>
      <resourceType1Name>
          <responseNameSatisfyingResourceType1/>
          <messageNameSatisfyingResourceType1/>
          <deleteResponseForResourceType1 action="delete"/>
          <addResponseForResourceType1 action="add"/>
      </resourceType1Name>
```

```
            <resourceType2Name>
                <modifyResponseForResourceType2 action="modify" />
                <messageNameSatisfyingResourceType2/>
                <deleteResponseForResourceType2 action="delete"/>
                <addResponseForResourceType2 action="add"/>
            </resourceType2Name>
        </statusUpdates>
            .
            .
</dataProviderName>
```

Figure 2-13 - Sample Configuration File Setup

diagram



type extension of **RequestType**

children **refId** **icdVersion** **username** **securityToken** **typeAStatus** **typeBStatus** **typeCStatus**

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	optional			
providerType	identifier	optional			

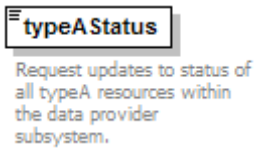
annotation documentation: A request sent by client to subscribe to data updates from the system

source

```
<xs:element name="subscribeReq">
 <xs:annotation>
  <xs:documentation>A request sent by client to subscribe to data updates from the system</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="RequestType">
    <xs:sequence>
     <xs:element name="typeAStatus" type="xs:boolean" default="false" minOccurs="0">
      <xs:annotation>
       <xs:documentation>Request updates to status of all typeA resources within the data provider subsystem.</xs:documentation>
      </xs:annotation>
     </xs:element>
     <xs:element name="typeBStatus" type="xs:boolean" default="false" minOccurs="0">
      <xs:annotation>
       <xs:documentation>Request updates to status of all typeB resources within the data provider subsystem.</xs:documentation>
      </xs:annotation>
     </xs:element>
     <xs:element name="typeCStatus" type="xs:boolean" default="false" minOccurs="0">
      <xs:annotation>
       <xs:documentation>Request updates to status of all typeC resources within the data provider subsystem.</xs:documentation>
      </xs:annotation>
     </xs:element>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

Figure 2-14 - subscribeReq Model

diagram



type **xs:boolean**

annotation documentation: Request updates to status of all typeA resources within the data provider subsystem.

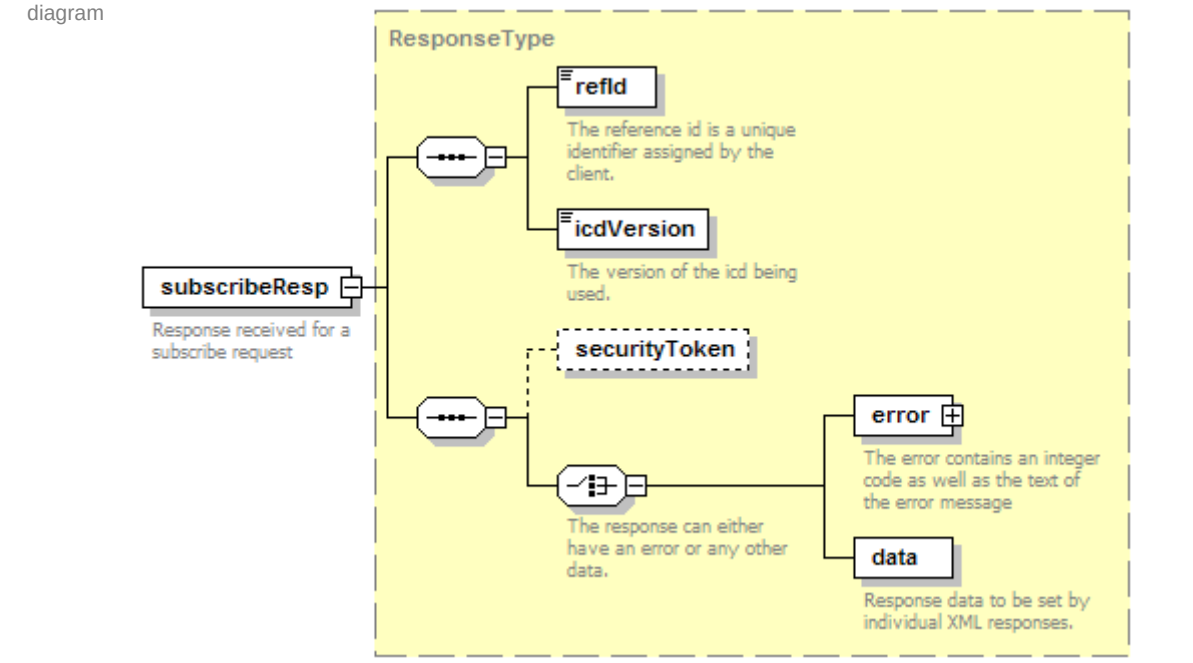
source

```
<xs:element name="typeAStatus" type="xs:boolean" default="false" minOccurs="0">
 <xs:annotation>
  <xs:documentation>Request updates to status of all typeA resources within the data provider subsystem.</xs:documentation>
 </xs:annotation>
</xs:element>
```

Figure 2-15 - Subscription Data Type Model

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.



type extension of **ResponseType**

children **refId** **icdVersion** **securityToken** **error** **data**

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	optional			
providerType	identifier	optional			

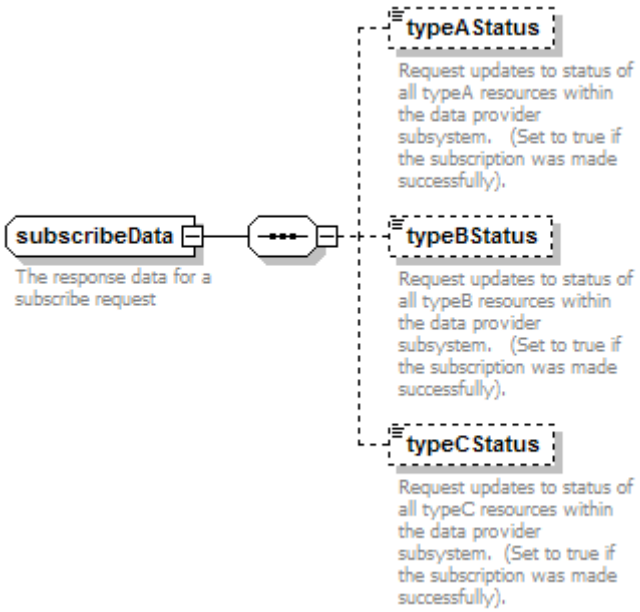
annotation documentation Response received for a subscribe request

source

```
<xs:element name="subscribeResp">
 <xs:annotation>
  <xs:documentation>Response received for a subscribe request</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="ResponseType"/>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

Figure 2-16 - subscribeResp Model

diagram



type extension of **responseData**

children **typeAStatus** **typeBStatus** **typeCStatus**

annotation documentation The response data for a subscribe request

source

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

Figure 2-17 - subscribeData Model

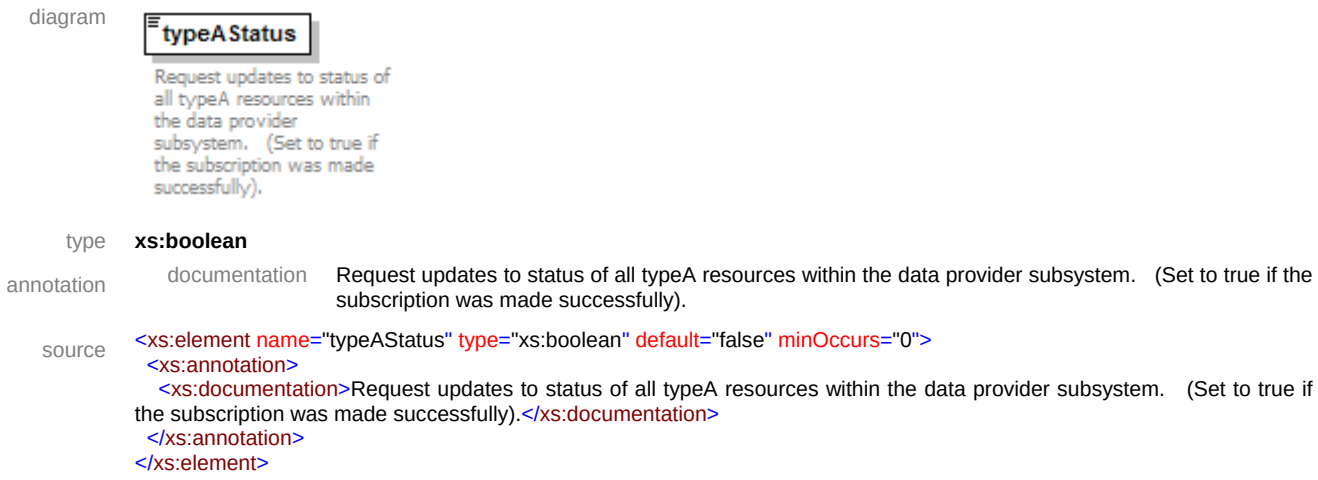


Figure 2-18 - Subscription Data Type Model

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 fomats). 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*.)

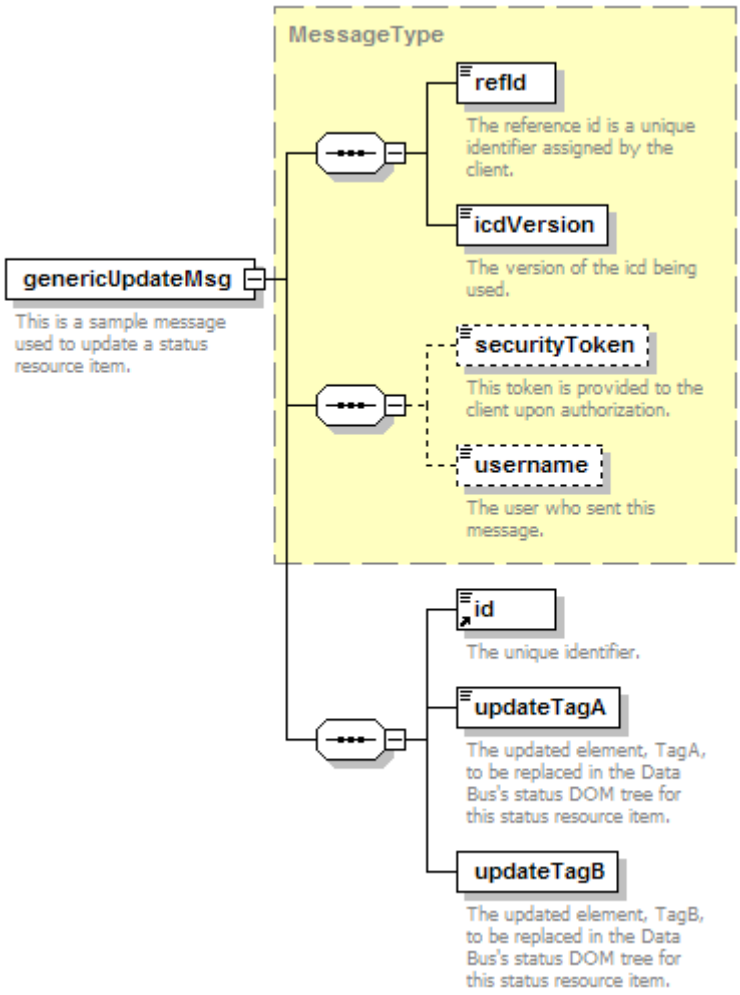
The Data Bus searches breadthfirst for an *id* element in the XML update message. Once this element is found, all elements listed in the same level of the XML document as the *id* element are considered to be status update tags that the Data Bus will attempt to replace in the pertinent status DOM tree.

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.

diagram



type extension of **MessageType**

children **refId** **icdVersion** **securityToken** **username** **id** **updateTagA** **updateTagB**

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	optional			
providerType	identifier	optional			

annotation documentation This is a sample message used to update a status resource item.

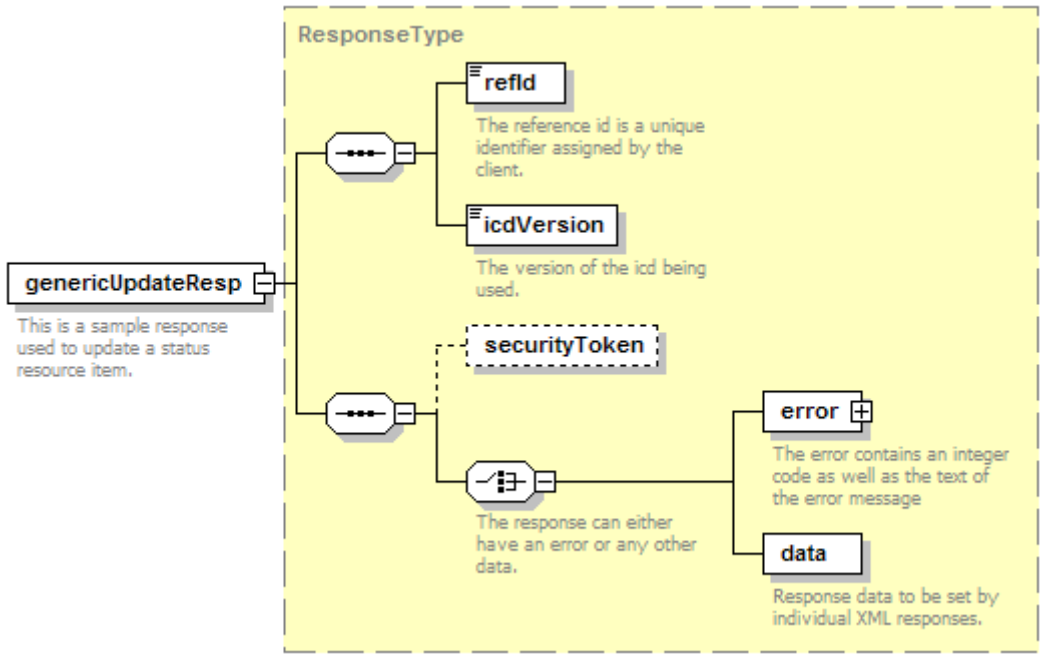
source

```
<xs:element name="genericUpdateMsg">
 <xs:annotation>
  <xs:documentation>This is a sample message used to update a status resource item.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="MessageType">
    <xs:sequence>
     <xs:element ref="id">
      <xs:annotation>
```

```
            <xs:documentation>The string identifying monitor to display the camera on.</xs:documentation>
          </xs:annotation>
        </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:sequence>
    </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

Figure 2-19 - Generic Update Message Model

diagram



type extension of **ResponseType**

children **refId** **icdVersion** **securityToken** **error** **data**

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	optional			
providerType	identifier	optional			

annotation
documentation This is a sample response used to update a status resource item.

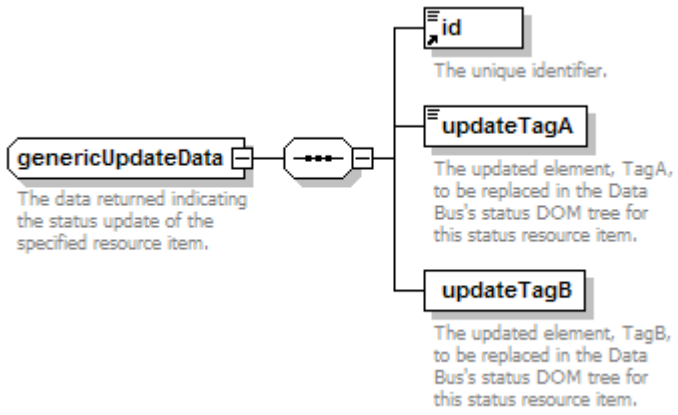
source

```
<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>
```

```
</xs:element>
```

Figure 2-20 - Generic Update Response Model

diagram



type extension of **responseData**

children **id** **updateTagA** **updateTagB**

annotation documentation The data returned indicating the status update of the specified resource item.

source

```
<xs:complexType name="genericUpdateData">
 <xs:annotation>
  <xs:documentation>The data returned indicating the status update of the specified resource item.</xs:documentation>
 </xs:annotation>
 <xs:complexContent>
  <xs:extension base="responseData">
   <xs:sequence>
    <xs:element ref="id">
     <xs:annotation>
      <xs:documentation>The id of the camera</xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="updateTagA" type="xs:boolean">
     <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:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

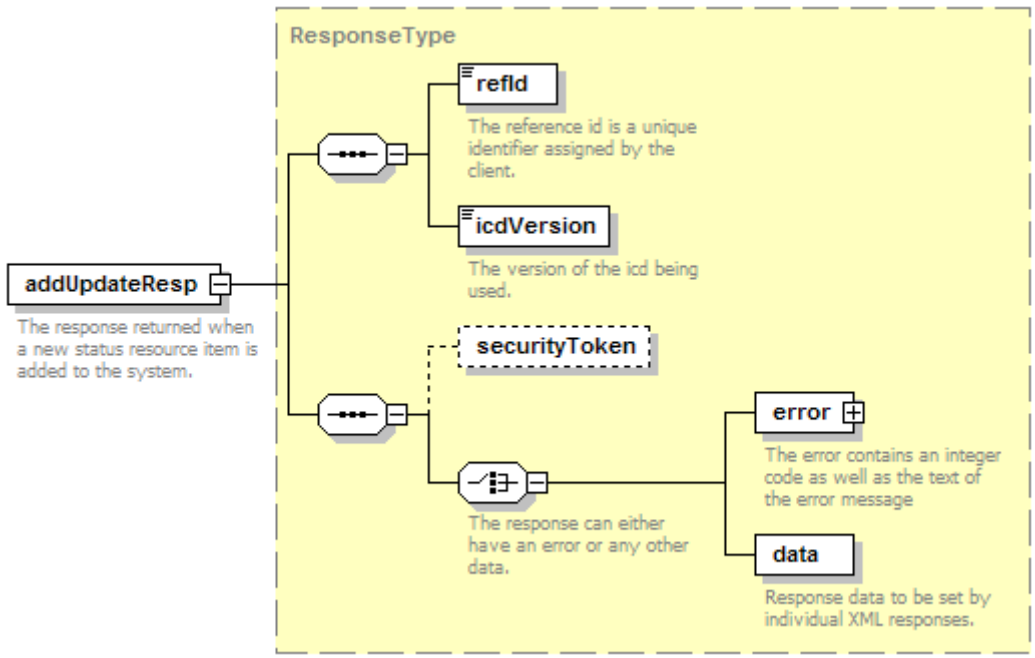
Figure 2-21 – genericUpdateData Model

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.). The

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.



type extension of **ResponseType**

children **refId** **icdVersion** **securityToken** **error** **data**

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	optional			
providerType	identifier	optional			

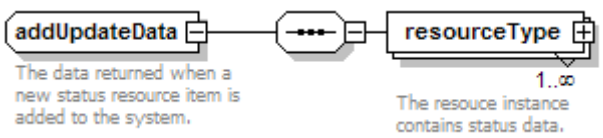
annotation documentation The response returned when a new status resource item is added to the system.

source

```
<xs:element name="addUpdateResp">
 <xs:annotation>
  <xs:documentation>The response returned when a new status resource item is added to the system.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="ResponseType"/>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

Figure 2-22 - Add Update Response Pattern Model

diagram



type extension of **responseData**

children **resourceType**

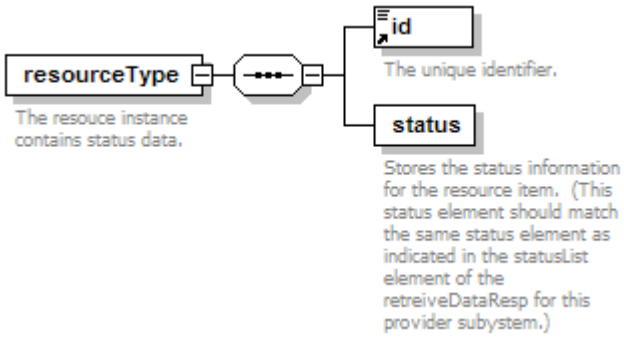
annotation documentation The data returned when a new status resource item is added to the system.

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 resouce instance contains status data.</xs:documentation>
     </xs:annotation>
     <xs:complexType>
      <xs:sequence>
       <xs:element ref="id"/>
       <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 retreiveDataResp for this provider subystem.)</xs:documentation>
        </xs:annotation>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Figure 2-23 – addUpdateData Model

diagram



children **id** **status**

annotation documentation The resouce instance contains status data.

source

```
<xs:element name="resourceType" maxOccurs="unbounded">
 <xs:annotation>
  <xs:documentation>The resouce instance contains status data.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="id"/>
```

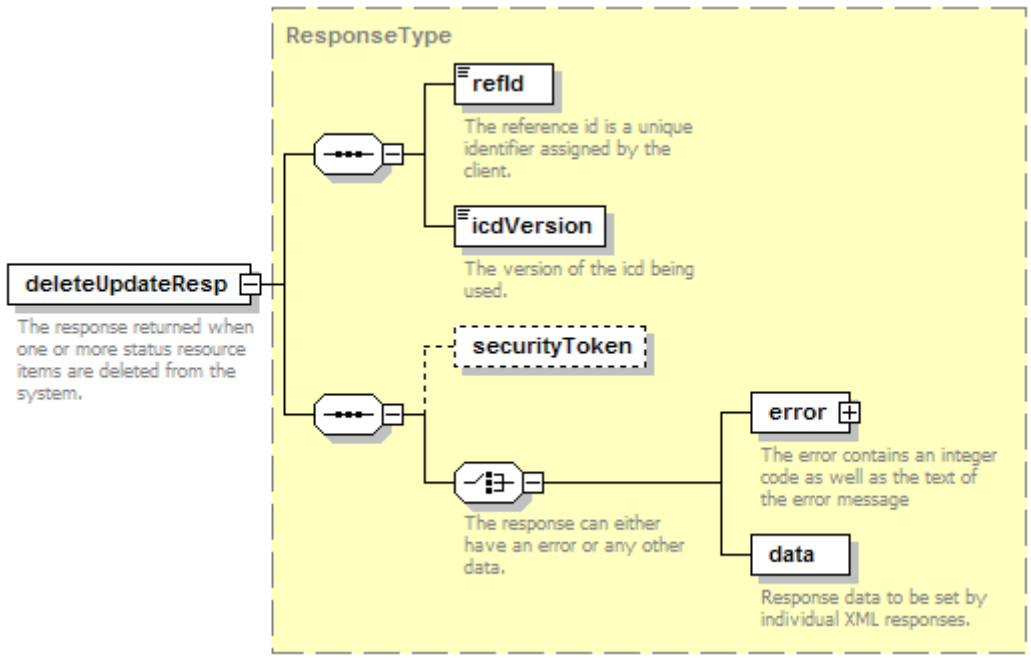
```
    <xs:element name="status">
      <xs:annotation>
        <xs:documentation>Stores the status information for the resource item.  (This status element should match the same status element as indicated in the statusList element of the retreiveDataResp for this provider subystem.)</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Figure 2-24 - resourceType 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).

diagram



type extension of **ResponseType**

children **refId** **icdVersion** **securityToken** **error** **data**

attributes

Name	Type	Use	Default	Fixed	Annotation
providerName	identifier	optional			
providerType	identifier	optional			

annotation documentation The response returned when one or more status resource items are deleted from the system.

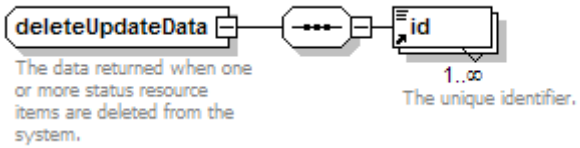
source

```
<xs:element name="deleteUpdateResp">
 <xs:annotation>
  <xs:documentation>The response returned when one or more status resource items are deleted from the system.</xs:documentation>
 </xs:annotation>
```

```
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="ResponseType"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

Figure 2-25 - Delete Update Response Pattern Model

diagram



type extension of **responseData**

children **id**

annotation documentation The data returned when one or more status resource items are deleted from the system.

source

```
<xs:complexType name="deleteUpdateData">
  <xs:annotation>
    <xs:documentation>The data returned when one or more status resource items are deleted from the system.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="responseData">
      <xs:sequence>
        <xs:element ref="id" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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.