Interface & Control Systems, Inc.

8945 Guilford Rd, Suite 120
Columbia, Maryland 21046
410.290.7600
www.sclrules.com

Spacecraft Markup Language ™

Draft Specification

Latest version:
Version 1.0.5 12/22/1999, Jay Offutt
Version 1.0.4 11/08/99 Pat Cappelaere
Version 1.0.3 10/29/1999, Jay Offutt

Principal Writers:
Pat Cappelaere, Jay Offutt (ICS, Inc.)

Editors:
Madeleine Marshall (ICS, Inc.)

Abstract

Spacecraft Markup Language (SML) is an extension of the Extensible Markup Language (XML) providing the Space Community with a standard definition of XML tags and concepts of structure to allow the definition of spacecraft and other support data objects. The SML reserved tags are defined within this document, as well as the requirements for data definition. The intent is to provide a language that allows the user extensive flexibility, while preserving cross-vendor compatibility with an open standard.

Table of Contents

1 Introduction

As space hardware becomes faster, space applications become more capable, and as a result also become more complex. With this complexity the definition of the command and telemetry databases required to monitor and control these applications, take more and more time to develop. Although every ground system vendor has their own proprietary tools and data formats to define these databases, there is a definite need for standardization to allow spacecraft and mission designers to design the control systems in a standard format. CCSDS and other packet definitions define protocols for passing data from one device to another, but the contents of those packets needs to be defined and described within a standard.

There have been a number of “languages” used to define the formats and contents of command and telemetry messages (or packets). One such language called “Record Definition Language” (RDL). (RDL was developed by GSFC for use in their ASIST ground system). RDL is a good starting point, but not accepted as a standard. One strong point for RDL is the use of ASCII to define the data items within packets for portability across platforms. RDL has the language provisions to clearly represent the record formats for commands and telemetry, but is limited in its ability to be extended. Another detriment is the fact that each vendor would need to write RDL compilers for their use. As RDL language evolves, and the RDL structures change.

This specification is intended to define a standard representation for describing the contents of the data streams that go to and from applications. These applications can be distributed within a network, across a continent, or between ground and space flight systems. The protocol used to transport the message is not addressed. There has been an underlying assumption that the data will be placed in packets of one kind or another for transportation. This specification proposes a space application extension to “Extensible Markup Language” (XML) for describing both the message definition (the schema), and the contents of specific messages (instances of messages) for commands, telemetry, archiving, and even inter-application communications. The constructs are largely based upon the structures provided in RDL, but are defined in the language itself. This approach provides greater flexibility when the user needs to extend the language, while still remaining conformant to the standard.

The Advent of Extensible Markup Language (XML)

In looking to the Internet, there is a new emerging standard for data representation. Extensible Markup Language (XML) has become the standard tool used to describe data in a human readable format that is well defined, and therefore also readable by computer software. As with other Internet protocols and standards, there are now a number of good quality tools freely available to develop and distribute XML documents. XML can be used to not only define the layout and content of messages (documents), but also to define instances of those messages too. Parsers and GUI viewers for XML are readily available on the Internet.

The web pages http://monet.astro.uiuc.edu/~dguillau/these/report_bits/XML.html and http://www.w3.org/XML/1999/XML-in-10-points provide basic information about XML and its uses.

Outside of the Space Community, XML is widely used to define domain specific languages such as MathML™, or the Microsoft® BizTalk™[1]. The BizTalk framework is an XML framework for application integration and electronic commerce. It includes a design framework for implementing an XML schema and a set of XML tags used in messages sent between applications. Microsoft Corp., other software companies and industry standards consortiums will use the BizTalk Framework to produce XML schemas in a consistent manner.

The following links point to XML vocabularies and applications of XML in various industries:
http://xdev.datachannel.com/directory/industry_resources.html
http://www.schema.net/

The Space Community has also started some efforts in using the technology with:

AML[2] The Astronomical Markup Language, developed by Damien Guillaume to describe astronomical objects.

AIML[3] The Astronomical Instrument Markup Language (AIML) is a domain-specific implementation of the more generalized Instrument Markup Language (IML).

Both AIML and IML are vocabularies based on the W3C standard, the Extensible Markup Language (XML). NASA Goddard Space Flight Center and Century Computing, a division of AppNet, Inc., are developing AIML to command and control astronomical instruments.

Since a number of groups have already started to define “vocabularies” to customize XML to their areas of expertise, the need for a standard vocabulary to be used by the

[1] http://www.biztalk.org/BizTalk/default.asp
[2] http://monet.astro.uiuc.edu/~dguillau/these/
[3] http://pioneer.gsfc.nasa.gov/public/aiml/

entire Space Community is even more urgent. The XML tags used by each group should be from the same definition so the data definition files and the resulting data files are interchangeable and exchangeable between the various vendors' software tools. Without a standard set of tag names, and XML document structure, each group will produce vocabularies, and grammars, that will again be proprietary to that single company or group.

2 Goals

This document is intended to specify a language definition, using XML as a base, which provides the following:

- A Standard representation language to define data objects
- User flexibility to define unique data structures
- A structure for defining data content in an object oriented fashion
- A language that is parsable by computer software
- A language that is readable by humans

3 "Spacecraft Markup Language" (SML)

The purpose of Spacecraft Markup Language (SML) is to extend XML to provide a standardized representation to be used in the space domain. The data that is being described and contained within SML can be commands, telemetry, abstract messages, science data, etc. The database to be defined by SML can contain some or all of the following: Commands, Telemetry Packets, Application messages, Events, Science data, status information for logging, etc. To create an orderly and clearly defined structure of the data to be processed, a definition of scope for the data objects can be defined that allow the data items to be processed by the software system using a protocol stack approach.

3.1 SML used to define a protocol stack

The SML structure provides an object-oriented way to describe the structure of a database. With this structure, a protocol stack can be implemented that allows the low level data "objects" to be encapsulated, or wrapped by outer layers. The wrapping of the data allows the user to constrain the development scope, leaving the details of data routing and data query information to an outer layer. SML can be used to define the layers such as the following.

Layer 0 is for low-level data that is exchanged on some physical transport layer. This layer could be the uplink/downlink, 1553/1394 Bus on the spacecraft or FIELDBUS used to interface instruments and sub-systems. In this layer, enough information has to be defined to support compression of the information for bandwidth-limited applications.

This specification does not address this layer other than to recognize the need for a transport layer, and to provide sufficient information to this layer that the data can be reliably encoded and decoded for transmission and receipt. A message provided to the input of this layer must be transferred, and reconstructed at the output of this layer without data loss.

Layer 1 is used to define the application messages. Application messages include what have historically been called commands, command packets, telemetry frames, telemetry packets etc. These messages can be exchanged over a software bus, point-to-point over TCP/IP or via a CORBA IIOP protocol. The protocol used for Layer 0 (transport layer) is not relevant to the definition of a message, as the message contents do not require any protocol information.

Layer 2 is used to represent processed information or science data as it is generated from the raw data. Data at this layer could be viewed as “files” or streams of information, or as collections of messages. The description of the content of the data file/stream shall be standardized to allow different vendor solutions the capability to exchange data without ambiguity. This layer can also be used to define the algorithms necessary for special processing of the science data.

Layer 3 is used for short-term storage of the information. The state of the application and any applicable state information (of outside agents to the system producing the data) could be represented at this layer. This layer shall define all the information needed for later querying.

Layer 4 is associated with long-term storage and retrieval of the information. The original physical hardware/software may or may not even be supported at the time of information retrieval.

As with other protocol stacks, during submission, each layer in this definition adds data to the inner (or lower) layer building up the message. As the data is later retrieved, the software dealing with each layer removes the outer layer information and the resulting data is passed to the next process.

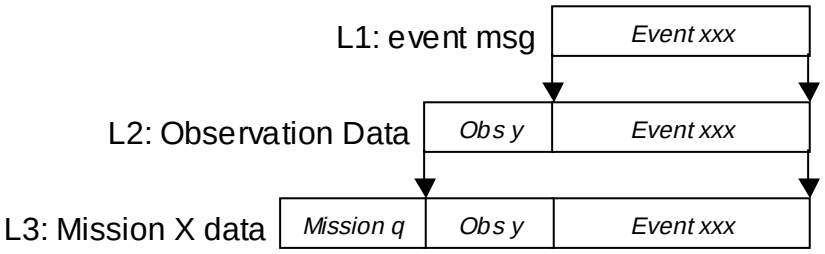


Figure 1- Example diagram for layer visualization

As Figure 1 shows, data that represents an event during a specific mission observation

is received and is contained within the full message structure. As these messages are stored in the short-term archive, additional state information is added to the message, and is passed to the archiver application. As the mission comes to an end, the mission specific header is attached to the observation data. The new (larger) message is sent to the long-term archiver for permanent storage.

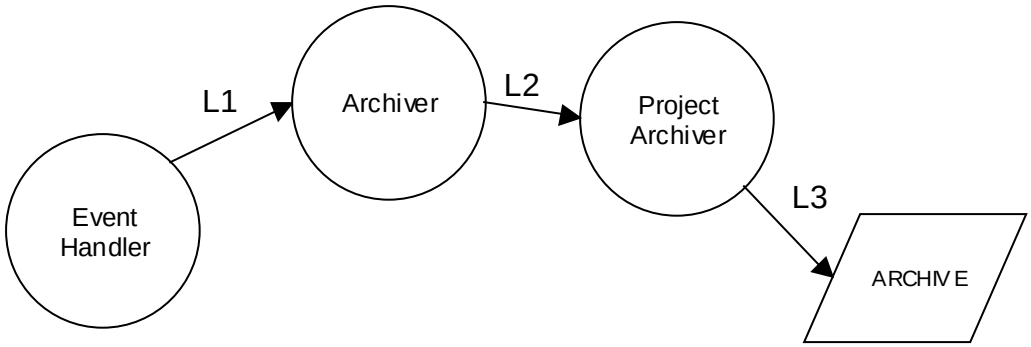


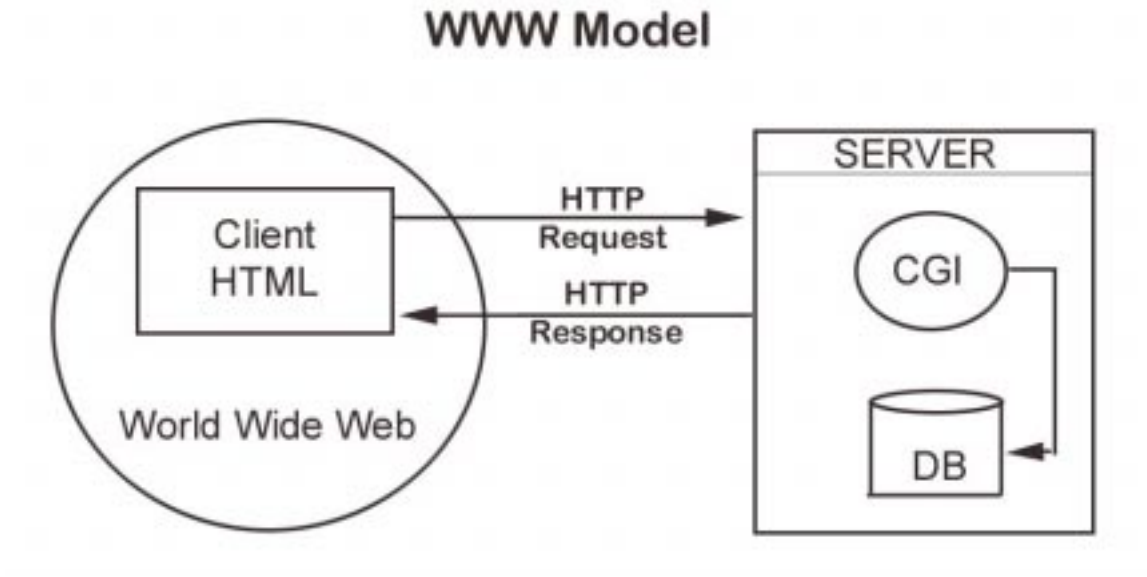
Figure 2- Process flow of data

This data flow shows the interaction of the layers. As the original data message passes through the processes, additional information is added to the message in a structured and well-defined manner. When the message is retrieved from the long-term archive, all the pertinent mission specific information is available directly with the “science content”.

4 Context of Operations

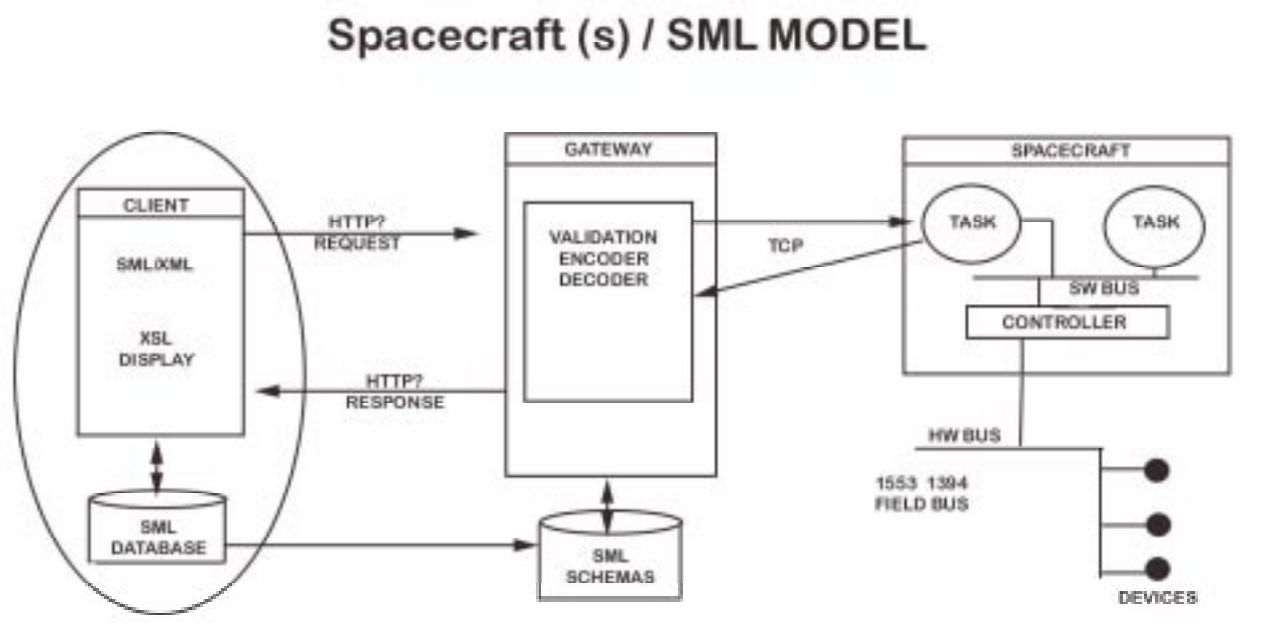
The traditional use of the Internet and the World Wide Web (WWW) is to use a web browser as a Graphical User Interface (GUI) to view information, and submit requests to information servers.

4.1 The Traditional World-Wide Web Model



4.2 Web Model for Space Operations

For space operations, the same web browser is used to present views of the SML information (for example commands) and allows the user to select and submit these objects to the Web Server (the gateway process) for transmission and processing.



In order to support the bandwidth-limited environment (spacecraft uplink/downlink) the gateway process encodes and decodes the data for transfer through “layer 1” (in the diagram called “TCP”). This is completely transparent to the user.

Here is a sequence for sending a command to the spacecraft.

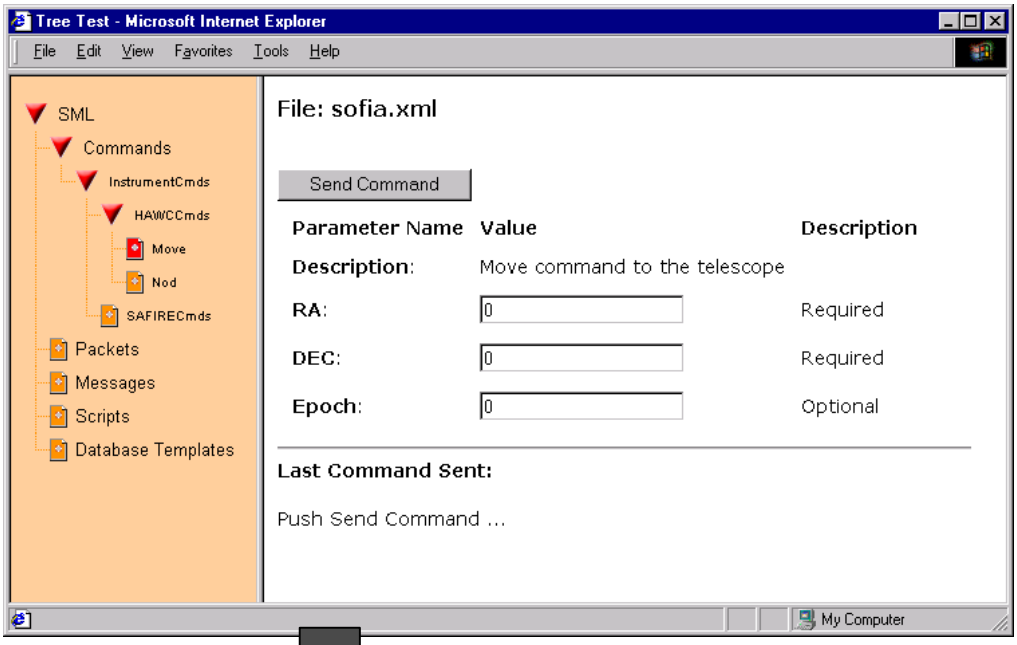
1. The user “sends” a command from the GUI interface (shown as CLIENT in the diagram). The GUI tool generates an SML message, and sends it to the Gateway (WEB Server).
2. The “SML” message is parsed, validated, and the contents of the message encoded to be transferred to the spacecraft.
3. The Spacecraft receives the encoded message, and passes to the destination “process”.
4. Onboard process operates on the message.

User Web Interface Generation

One of the benefits of using an XML derivative to define the database items, is the available tools for building and displaying the data. The resulting browser based tool allows the user to view the database, and even send messages and commands using standard web browser tools.

4.2.1.1 Example HAWC Move Command

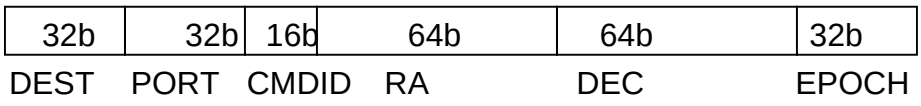
As the user selects a command from the database view, the browser presents (again using a Style Sheet) a data entry form to provide the parameters to the command. As the Submit button is pressed, the script forms an SML “sheet” and sends the sheet to the Gateway Web Server. The gateway takes the SML and encodes the data into a binary packet for transmission to the spacecraft.



XML Command Message

```
<CMD NAME="Move">
    <DEST>23.1.3.4</DEST>
    <PORT>2001</PORT>
    <CMDID>45</CMDID>
    <RA>12:14:44.5</RA>
    <DEC>0:00:00.0</DEC>
    <Epoch>0000.0</Epoch>
</CMD>
```

Encoding at Uplink



Concept of Operations

Knowledge engineers use SML to define application specific data and meta-data. In the process, a meta-schema is used to define the hierarchy or structure of all the objects.

An SML builder can be used to build the meta-schema and populate the objects. The ASCII XML file is what is being placed under configuration management and used as a data interchange medium.

Several other tools can be developed based on the existence of XML tools such as parsers, editors to build a command editor/browser, a telemetry viewer and archive viewer…

5 SML-Level 1 Language Reference

5.1 Overview/Concepts

The definitions are based on a hierarchical structure. The structured tree can be a container tree made of database and section tags or can be a hierarchy tree made of a schema and derived elements.

The SML is used to define various objects such as commands, telemetry packets, and messages in a hierarchical manner. Object-oriented methodology can be used to define and further refine the definitions. Packets or messages can be viewed as sequences of elements. The user can assemble those pre-defined or user derived elements to form a packet, command or message.
As the user defines objects or data relevant to an application, an object-oriented approach is used to break the problem into manageable sub-components.
SML provides the capability to define schema and data. The schema definition capability allows the user to define structures of data and define what properties are allowed to be specified for the data structures. The data is then defined using the fields (tags) that are defined by the schema.

Figure 3- Schema Hierarchy Tree, shows representation of the Hierarchy of the above example as viewed as a tree structure:

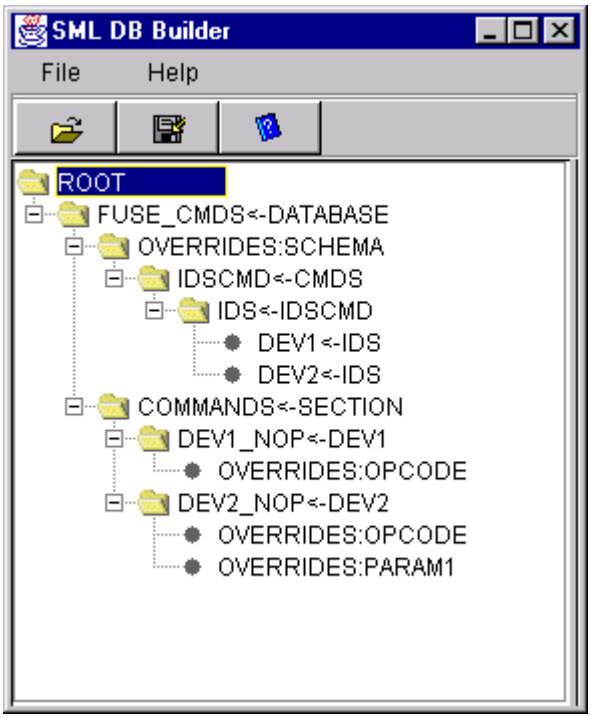


Figure 3- Schema Hierarchy Tree

Below is the text version of the example SML shown in Figure 3. This text version shows the definition of two command messages and the class hierarchy or schema, that defines all sub-components of the commands.

```
<?xml version=”1.0” standalone=”NO” ?>

<DATABASE NAME="FUSE_CMDS" xmnls="x-schema:sml.xml">
     <SCHEMA>
          <CMDS NAME=”IDSCMD” DESC=”IDS Commands” >
               <UB NAME=”FCTN”/>
          </CMDS>

          <IDSCMD NAME=”IDS” DESC=”IDS Commands” >
               <UB NAME=”OPCODE”> 0 </UB>
          </IDSCMD>

          <IDS NAME=” DEV1” DESC=”IDS Commands” >
               <FCTN> 10 </FCTN>
          </IDS>

          <IDS NAME=” DEV2” DESC=”IDS Commands” >
               <FCTN> 11 </FCTN>
               <UI NAME=”PARAM1” />
          </IDS>

     </SCHEMA>
    <SECTION NAME=”COMMANDS”>

          <DEV1 NAME=”DEV1_NOP”>
               <OPCODE> 0 </OPCODE>
          </DEV1>

          <DEV2 NAME=”DEV2_NOP”>
               <OPCODE> 0 </OPCODE>
               <PARAM1> 0 </PARAM1>
          </DEV2>

    </SECTION>
</DATABASE>
```

First, some nomenclature, given this piece of the above example:

```
<IDSCMD NAME=”IDS” DESC=”IDS Commands” >
        <UB NAME=”OPCODE”> 0 </UB>
</IDSCMD>
```

The “TAG” is “IDSCMD”.
The “NAME PROPERTY” (portion of text with NAME=”IDS”) is IDS. (“NAME” is the name of the property, and IDS is it’s value).
The DESC attrubute has the value “IDS Commands”. (“DESC” is the name of the property, and “IDS Commands” is it’s value).
The line <UB NAME=”OPCODE”> 0 </UB> indicates that this object has a field that is a “UB” (an unsigned byte object), with name OPCODE, and sets the value of this field to 0.

Reference file example shown above:

The format of this SML file has a number of “Containers” that are identified by matched “TAGS”. *XML TAGS are represented by <name> as the opening TAG called “name”, and the closing TAG </name>.* The outermost container is the “DATABASE” container. The DATABASE is named “FUSE_CMDS”, and in this case has a description property. The DATABASE named FUSE_CMDS contains a SCHEMA, and one SECTION. The SCHEMA contains the definition of 4 command objects. These 4 objects form a hierarchy. The lineage of each object is traced by looking at the object name property (NAME=xxx), and the object’s TAG name <xxx>. The TAG holds the name of the object’s parent object, and the NAME= property holds the new object name. For example, the new object defined with the following:

```
<IDSCMD NAME=”IDS” DESC=”IDS Commands” >
         <UB NAME=”OPCODE”> 0 </UB>
</IDSCMD>
```

is named “IDS”, and is based upon the structure of it’s parent that is named “IDSCMD”. This object “IDS” will inherit all fields that IDSCMD has, and in this case is adding a new field of type UB (unsigned byte) named “OPCODE”, and setting the default value for that field to 0.

Following the object definitions in the SCHEMA section, there is the opening “SECTION” tag with NAME property “COMMANDS”. Within the opening and closing SECTION tag are two command definitions. These commands have parents that were defined in the schema section. As an example, the class hierarchy for the DEV2_NOP command is defined as follows:

DEV2_NOP extends a DEV2 object (in this case it represents a command). It sets the field “OPCODE” to 0, and sets field “PARAM1” to 0. Both fields OPCODE, and PARAM1 are inherited from the parent object DEV2. DEV2_NOP also has a FCTN field that is set to 11 by the parent object DEV2. DEV2 extends an IDS object. The DEV2 object inherits a field “FCTN” from the IDS object, and sets the value to 11. DEV2 also adds a new field called PARAM1 and defines it as an UI (Unsigned Integer).

5.2 SML File Structure

The general SML file structure is:

```
HEADER
<DATABASE>
        <SECTION>
                <OBJECTS>
                </OBJECTS>
        </SECTION>
</DATABASE>
```

5.3 File Header

Format

```
<?xml version=”1.0” standalone=”NO” ?>
```

This header is required for xml processing.

A root level tag is also required for each file. The root level tag requires access to the SML definitions using the namespace definitions.

```
<DATABASE NAME="FUSE_CMDS" >
```

Grouping Constructs

Note: XML nesting is used by containers to define “has-a” relationships. It does not imply an inheritance or “is-a” relationship.

Before populating a database, a user is very likely to define or extend an existing SML meta-schema. The next step would then be to define a database. The database could be broken in various sections.

5.3.1.1 Database Tag

A database is a container that contains SML objects.

Format

```
<DATABASE NAME=””
```

```
        [DESC=””] [VERSION=””] [xmlns:””]>
    </DATABASE>
```

This is a top level node of the tree. It has to contain namespace definitions.

Example:

```
<DATABASE NAME = ”FUSE”
     DESC= ”FUSE Database Definition”
     VERSION = ”version 3.5” >

</DATABASE>
```

Properties	***Required***	***Description***
NAME	*Yes*	
VERSION	*No*	
DESC	*No*	
xmnls	*Yes*	*Namespace definition. See XML namespaces*

5.3.1.2 Schema Tag

A Schema is a root node for deriving new classes. Those classes will be used to populate a database. This is a user convenience tag. This tag is generally used when defining an object oriented commanding system or when defining a database structure. See examples in the appendices.

Format

```
<SCHEMA NAME=”” [DESC=””] [VERSION=””]>
</SCHEMA
```

Example:

```
<SCHEMA = ”FUSE_CMDS”
    DESC = ”FUSE Command Definition”
    VERSION  = ”version 3.5” >
</SCHEMA>
```

Properties	*Required*	*Description*
NAME	*No*	*Name of the SCHEMA*
VERSION	*No*	*Version of the SCHEMA*
DESC	*No*	*A text description of the SCHEMA*

5.3.1.3 Section Tag

A database can be partitioned in various sections. Sections are containers and can be nested. This is a user convenience tag. Database and sections can be viewed as containers use to segment the information similar to what one would do within an outline document. The intent of SECTIONS is to allow the user to break the database objects into manageable sized groupings of related objects.

Format

```
<SECTION NAME="" [DESC=""]>
</SECTION>
```

Example:

```
<SECTION NAME="SCL Compiler messages">
     <SECTION NAME="Input Messages" >
     </SECTION>
</SECTION>
```

Properties	*Required*	*Description*
NAME	*Yes*	*Text – name of the section*
DESC	*No*	*Test – description of the section*

Primitive Element Types

Format

```
<data_types> </data_types>
```

Data_types

Type	Size	Description
UB	*1*	*Unsigned Byte*
SB	*1*	*Signed Byte*
UI	*2*	*Unsigned short integer*
SI	*2*	*Signed short integer*
ULI	*4*	*Unsigned long integer*
SLI	*4*	*Signed short integer*
FP	*4*	*Single Floating Point*
DFP	*8*	*Double Floating Point*
TIME	*4*	*GMT Standard Time*
STRING	*N*	*Variable size String*
CHAR	*N*	*Fixed size string buffer. Use SIZE property*

Properties

Properties	***Requir***	***Description***
NAME	*Yes*	*Text – the name of the data type*
DESC	*No*	*Text – a description of the data type*
MODE	*No*	*Text - Options="STATIC,DYNAMIC"*

MASK	*No*	*Text - Decimal, Hex value: 0xFFFF or Binary Value: 1xFFFF*
MIN	*No*	*Number – the minimum value of the data type*
MAX	*No*	*Number – the Maximum value for the data type*
BIGENDIAN	*No*	*Boolean, default=yes*
UNITS	*No*	*Text – User defined*
VISIBLE	*No*	*Boolean – Element to be displayed on screen. Default equals true*
RHIGH	*No*	*XFP – For analog Red Limit*
YHIGH	*No*	*XFP – For analog Red Limit*
YLOW	*No*	*XFP – For analog Red Limit*
RLOW	*No*	*XFP – For analog Red Limit*
DEFAULT	*No*	*Default value*
AS	*No*	*String name of element to be replaced*
DRANGE	*No*	*Text – name of discrete range*
METHOD	*No*	*Text – name of a Method to use*

Auxiliary properties

DRANGE: this element is used for discrete fields. It specifies the various states in terms of values and state names.

```
<DRANGE>
      <OPTION VALUE=”1” NAME=”OPEN” />
      <OPTION VALUE=”0” NAME=”CLOSE” />
</DRANGE>
```

The METHOD element is used when special processing is required. It can be used for instance to compute a checksum or convert raw values to engineering values.

<METHOD NAME="POLYS" PARMS=”0 1 2.1 />

Packet Tag

This container tag is used to define telemetry packets. A packet contains elements. Each element has a defined name and type. Elements are defined later in this document.

Format

```
<PACKET NAME=”” [DESC=””]>
</PACKET>
```

Example:

```
<PACKET NAME=”hdr”
        DESC=”define the packet header” >
    <UB NAME=”byte1” />
    <UI NAME=”int1” />
</PACKETS>
```

Properties	***Required***	***Description***
NAME	*Yes*	*Text – name of the Packet*
DESC	*No*	*Text – description of the Packet*

Commands (“CMDS”) Tag

This tag is used to define command classes. The command tag is a container. It is defined as a sequence of primitive elements that contain enough information to encode the packet into the proper bit pattern. The command definition also contains enough information to support a user interface and validate user entries against expected range.

Format

```
<CMDS NAME=”” [DESC=””]>
</CMDS>
```

Example

```
<CMDS NAME=”MOVE” >
     <UI NAME=”ADDR” />
     <UI NAME=”PORT” />
     <DFP NAME=”RA” />
     <DFP NAME=”DEC” />
</CMDS>
```

Properties	***Required***	***Description***
NAME	*Yes*	*Text – name of the Command*
DESC	*No*	*Text – description of the Command*
ACCESS	*No*	*This option is used to define command grouping. Users will have access to the command depending on the access level assigned to the user.* *Possible options could be: ”NORMAL,CRITICAL,HAZARDOUS”* *DEFAULT NONE*
ABSTRACT	*No*	*Boolean; abstract class—cannot be instantiated* *DEFAULT FALSE*

5.3.1.4 Protocol support tags

Header and trailer tags have been reserved for protocol definition. Trailer information will be added to the end of the command and/or derived commands from that parent class. Header definition is usually omitted.

Example:

```
<CMDS NAME=”CCSDS_CMD” >
      <HEADER>

      </HEADER>

      <TRAILER>
          <UI NAME=”Checksum” …/>
      </TRAILER>
</CMDS>
```

Header Tag

The Header tag is used an optional tag used to define special protocol information. This information will be inherited by all derived commands.

Format:

```
<HEADER [DESC=””] >
</HEADER>
```

Properties	***Required***	***Description***
DESC	*No*	*Text – description of the header*

Trailer Tag

The trailer information will be added to the command and its children if the command has been derived.

Format

```
<TRAILER> </TRAILER>
```

Properties	*Required*	*Description*
DESC	*No*	*Text – description of the trailer*

Aggregate Data Types

Aggregate data types allow you to define more complicated sequences of data, including switches (UNION), series of bits (BITFIELD), and groups of sequential data items (RECORD), array of items (ARRAY).

5.3.1.5 Record Tag

Defines a sequence of fields. These sequences can then be reused by name.

Format

```
<RECORD NAME=”” [DESC=””]>
</RECORD>
```

Example

```
<RECORD NAME="HDR1" >
    <UI NAME="Version" DEFAULT="0" />
    <UI NAME="Type" DEFAULT="1" />
</RECORD>
```

Properties	*Required*	*Description*
NAME	*Yes*	*Text : name of the Record*
DESC	*No*	*Text : description of the Record*
ID	*No*	*Integer: Used when database records are defined*
AS	*No*	*Previous Element Name, Used to substitute the element*

5.3.1.6 ARRAY Tag

A user can define an array of records or defined types by specifying a length property.

Format

```
<ARRAY NAME="" [DESC=""]>
</ARRAY>
```

Example

```
<RECORD NAME="MYREC">
    <UI NAME="recordID"/>
    <UI NAME="dataStructure"/>
    <UI NAME="pointerIncrement"/>
    <UI NAME="mask"/>
    <UI NAME="lshift"/>
    <UI NAME="order"/>
</RECORD>

<ARRAY NAME="item" LENGTH="3">

  <MYREC />

</ARRAY>
```

Properties	***Required***	***Description***
NAME	*Yes*	*Text – name of the Array*
DESC	*No*	*Text – description of the Array*
LENGTH	*No*	*Integer - Array length (number of elements)*

5.3.1.7 Union Tag

A UNION statement is used to map several elements onto the same data field. It is used when the user has to make a choice between various elements. When used in a telemetry packet, a union can indicate overlapping fields.

Format:

```
<UNION NAME="" [DESC=""]>
</UNION>
```

Example:

```
<UNION NAME="RNGPWST">
     <UI NAME="OFF" DESC="PWR OFF">2</UI>
     <UI NAME="ON" DESC="PWR ON">1</UI>
</UNION>
```

Properties	*Required*	*Description*
NAME	*Yes*	*Text – name of the Union*
DESC	*No*	*Text – description of the Union*
AS	*No*	*Text - Previous Element Name to be substituted*

5.3.1.8 Bitfield Tag

A BITFIELD defines a data field bit-by-bit. Like a UNION, all parameters within a BITFIELD are mapped to the same location (offset) within the command. But unlike a UNION, all parameters within the BITFIELD must be specified. When used in a telemetry packet, the behavior is very similar to the union.

Format

```
<BITFIELD NAME="" [DESC=""]>
</BITFIELD>
```

Example

```
<CMDS NAME="SAMPLE" DESC="SAMPLE CMDS">
     <BITFIELD NAME="WORD1">
     <UI NAME="FIRST" MASK="0xF000" />
     <UI NAME="SECOND" MASK="0x0F00" />
     <UI NAME="THIRD" MASK="0x00F0" />
     <UI NAME="FOURTH" MASK="0x000F" />
     </BITFIELD>
</CMDS>
```

Properties	***Required***	***Description***
NAME	*Yes*	*Text - name of the Bitfield*
DESC	*No*	*Text - description of the Bitfield*
AS	*No*	*Text - String Name of element to be replaced*

5.3.1.9 User Defined Types

TBD

Adding User Defined Properties

TBD

Object-Oriented Design

Meta-data definition should follow an object-oriented approach. For instance, commands can be defined using a top-down approach from generic commands to more specific commands.

Inheritance – Subtyping

SML does not make any difference between a class and a named object. A new class or derived class is a named object of the parent class with additional elements defined. Inheritance is implicit by using the tag name of the parent class.

Example

```
<CMDS NAME=”InstrumentCMds” >
      <UI name=”CMDID” />
</CMDS>

<!—HAWCCmds derived command from InstrumentCmd -->

<InstrumentCmds NAME= “HAWCCMds” >
      <ADDR>12.2.3.4</ADDR>
      <PORT>2002</PORT>
</InstrumentCmds>

<!—Definition of all HAWC Commands -->
<HAWCCmds NAME=”MOVE” >
      <DFP NAME=”RA” >
      <DFP NAME=”DEC” >
</HAWCCmds>
```

5.3.1.10 Default value override

The HAWCCmds specifies some default values for some elements.

Meta-Schema Modification

Elements can be replaced by new elements using the “AS” property. This allows the user to reuse an element but rename it with something more meaningful.
The previous element is not accessible anymore.

```
<UNION NAME=”FES” AS=“FCTN”>
      <UB NAME=”FESA” DEFAULT=”42” />
      <UB NAME=”FESB” DEFAULT=”43” />
</UNION>
```

This example allows the user to address the function code element as a “FES” element and offers the user a choice of FESA or FESB.

Properties of existing elements can also be changed using the “MODIFY” tag.

```
<MODIFY NAME=”FILTER” MIN=23 MAX=54 />
```

5.3.1.11 Scripting Support

To define scripting blocks that allow an end-user to interface with a scripting environment, execute scripts and pass parameters. This definition can be used by the viewer. It allows a user to select a particular script to execute, enter the parameters and submit the request to the server.

```
<SCRIPT NAME=”Maneuver1” DESC=”execute maneuver one”
      LANGUAGE=”SCL” SOURCE=”man.scl” >

      <UI NAME=”Parameter1” />
      <UI NAME=”Parameter2” />
</SCRIPT>
```

5.4 SML-Layer 1

This is to support application message definitions.

Notification Message

This is a one-way message. The application drops that message on the software bus.

Format

```
<MESSAGE NAME="" [DESC=""]>
    <NOTIFY>
    </NOTIFY>
</MESSAGE>
```

Example

```
<MESSAGE NAME="myMsg", DESC="my message" >
    <NOTIFY>
         <STRING name="filename" />
         ….
    </NOTIFY>
</MESSAGE>
```

Properties	***Required***	***Description***
NAME	*Yes*	*Text - name of the Message*
DESC	*No*	*Text - description of the Message*

Request Message Definition

This section addresses synchronous messaging between two applications. We are assuming that one application has registered as a handler for that particular message name. Applications can make the request on the software bus and the reply will come back from the handler as defined below.

Format

```
<MESSAGE NAME="" [DESC=""]>
    <REQUEST>
    </REQUEST>
    <REPLY>
    </REPLY>
</MESSAGE>
```

Example

```
<MESSAGE NAME=”myMsg” DESC=”my message” >
    <REQUEST>
         <STRING NAME=”filename”>
    </REQUEST>
    <REPLY>
         <UB NAME=”status” />
         <STRING NAME=”errstr” />
    </REPLY>
</MESSAGE>
```

Properties	***Required***	***Description***
NAME	*Yes*	*Text - name of the Message*
DESC	*No*	*Text - description of the Message*

Message Routing Headers

5.4.1.1 Routing Information

External Header

Biztalk routing format:

```
<ROUTE
    <FROM
         locationID="111111111"
         locationType="DUNS"
         process=""
         route="" handle="45"/>

    <TO
         <locationID="222222222"
         locationType="DUNS"
         process=""
         route=""
         handle="93"/>

</ROUTE>
```

Internal Bus Headers

```
<HEADER>
</HEADER>
```

5.4.1.2 Message Definitions

See SML Message Specification Document

5.5 Science Data Representation

Express AML in SML as example: http://monet.astro.uiuc.edu/~dguillau/these/

Express Pipelining (Algorithms used for data processing) in AIML (GSFC/Century Computing)

Astronomical Cataloguing: http://vizier.u-strasbg.fr/doc/astroxml.htx

Several science data and archive file formats have already been defined such as:
FITS: http://www.gsfc.nasa.gov/astro/fits/fits_home.html
HDF5: http://hdf.ncsa.uiuc.edu/HDF5/doc/ddl.html
CDF: http://nssdc.gsfc.nasa.gov/cdf/

A possible approach would be to take one and XMLize it as a strawman approach.

5.6 SML-Layer 3 – Short Term Storage

5.7 Data Logging

Example:

```
<?xml version="1.0" standalone="no" ?>

<LOGFILE
      NAME="fuse_log.xml"
      CREATED="12/12/12 12:12:12.00"
      xmnls="x-schema:sml.xml" >

</LOGFILE>
```

Properties	***Required***	***Description***
NAME	*Yes*	*Text - name of the LogFile*
DESC	*No*	*Text - description of the LogFile*

File Chaining tags:

This tag is used to chain log files when the maximum size has been reached.

```
<CHAIN
      TIME="12/12/1999 12:12:12.00"
      FILENAME="file.2.dat"
      SEQNUM="2"
      SIZE="1000" />
```

Properties	***Required***	***Description***
NAME	*No*	*Text - name of the CHAIN*
DESC	*No*	*Text - description of the CHAIN*
TIME	*Yes*	*Text – time of the creation of linked log file*
FILENAME	*Yes*	*Text – Name of the log file*
SEQNUM	*No*	*Text – Sequence Number*
SIZE	*No*	*Integer – number of bytes to break log files into in units of K Bytes.*

Log Records

A logfile can contain information, warning or alert messages. Those messages specify a time, a source and the data.

Information, Warning & Alert Messages

```
<INFO TIME="12/12/1999 12:12:12.00" SOURCE="SCLWEB">
</INFO>

<WARN TIME="12/12/1999 12:12:12.00" SOURCE="SCLWEB">
</WARN>

<ALRT TIME="12/12/1999 12:12:12.00" SOURCE="SCLWEB">
</ALRT>
```

Properties	***Required***	***Description***
NAME	*No*	*Text - name of the INFO item*
DESC	*No*	*Text - description of the INFO Item*

TIME	*Yes*	*Text – Time of the event*
SOURCE	*Yes*	*Text – The source process of the event*

Logging of SML Objects

Any SML object can be logged with the SML tags.

```
<SML TIME="12/12/1999 12:12:12.00" SOURCE="SCLWEB">
      <SCRIPT NAME="scrub">
          <PARAM1>12</PARAM1>
          <PARAM2>23</PARAM2>
      </SCRIPT>
</SML>
```

Logging of User Data Objects

User defined data can be logged with the data tag.

```
<DATA NAME=""
      TIME="12/12/1999 12:12:12.00" SOURCE="SCLWEB" >
<DATA>
```

5.8 SCL-Layer 4 – Long term storage

TBD.

5.9 Preprocessing Directives

These tags are processed using XSL.

```
<INCLUDE FILE=”filename” />


<DEFINE NAME=”XX” />

<IFDEF NAME=”flight”>
</IFDEF>
```

```
<IFNDEF NAME=”ground”>
</IFNDEF>
```

6 Meta-Schema Development

6.1 Object

An object is defined by a tag and some user data and an end-tag.

Example:

```
<OBJECT>data</OBJECT>
```

6.2 Properties

An object has what we will call in this context **properties.** One could define a TYPE property used to encode/decode/validate the user data field

Example:

```
<OBJECT TYPE=”string”>data</OBJECT>
```

6.3 Default Properties

An object has some default built-in properties:

Property	***Require***	***Description***
NAME	Yes	string
DESC	No	string
TYPE	No	String – See default types
DEFAULT	No	Depends - Default data value
REQUIRED	No	Boolean – default no
VISIBLE	No	Boolean (Is that property visible in the builder)

6.4 Derived Object

One can derive an object and generate a new tag.

Example:

```
<OBJECT NAME=”NEWOBJ”
     DESC=”derived New Object” TYPE=”int” />
```

This new object can now be used:

```
<NEWOBJ>23</NEWOBJ>
```

6.5 Adding Properties

One can add properties to the default built-in properties of an object by adding property object to the property list of that object.

Example:

```
<OBJECT NAME=”VOLTAGE”
     DESC=”Voltage Object” TYPE=”float”>
     <PROPERTY NAME=”UNITS” TYPE=”STRING”/>
</OBJECT>
```

The data field of that new property can be set within the tag field:

```
<VOLTAGE UNITS=”volts”>35.1</VOLTAGE>
```

6.6 Object fields

In addition to properties, objects can have fields. A field is a special object that has some specific properties.

Example:

```
 <OBJECT NAME=”FIELD” TYPE=”string” DESC=”Field Object”>
          <PROPERTY NAME=”MODE”/>
          <PROPERTY NAME=”NBITS” />
          <PROPERTY NAME=”SIZE” /> …
</OBJECT>
```

We can then derive an INTEGER and a REAL field and then specialize those fields with additional properties. We have current defined the following fields:

Fields	Description

UB	Unsigned Byte
SB	Signed Byte
UI	Unsigned short integer
SI	Signed short integer
ULI	Unsigned long integer
SLI	Signed short integer
SFP	Single Floating Point
DFP	Double Floating Point
TIME	GMT Standard Time
STRING	Variable size String
CHAR	Fixed size string buffer. Use SIZE attribute

They all have a similar property list:

Properties

Properties	***Requir***	***Description***
NAME	*Yes*	*Text – the name of the data type*
DESC	*No*	*Text – a description of the data type*
MODE	*No*	*Text - Options="STATIC,DYNAMIC"*
MASK	*No*	*Text - Decimal, Hex value: 0xFFFF or Binary Value: 1xFFFF*
MIN	*No*	*Number – the minimum value of the data type*
MAX	*No*	*Number – the Maximum value for the data type*
BIGENDIAN	*No*	*Boolean, default=yes*
UNITS	*No*	*Text – User defined*
VISIBLE	*No*	*Boolean – Element to be displayed on screen. Default equals true*
RHIGH	*No*	*XFP – For analog Red Limit*
YHIGH	*No*	*XFP – For analog Red Limit*

YLOW	*No*	*XFP – For analog Red Limit*
RLOW	*No*	*XFP – For analog Red Limit*
DEFAULT	*No*	*Default value*
AS	*No*	*String name of element to be replaced*
DRANGE	*No*	*Text – name of discrete range*
METHOD	*No*	*Text – name of a Method to use*

6.7 Containers

An object can contain fields (really derived objects). We need a way to specify what an object can contain. This information is very important to the GUI to determine what can be dragged and dropped in the object container. We can now say that an object has a list of properties, a list of what it can contains and a list of fields that it will eventually contain.
We use a special object to define the “contains” relationship. An object can have more than one “contains” object in its list.

Example:

```
<OBJECT NAME=”BAG”>
     <CONTAINS NAME=”TOMATOES” />
     <CONTAINS NAME=”BREAD” />
     <CONTAINS NAME=”BUTTER” />
</OBJECT>
```

6.8 Aggregate Fields

We can now define aggregate fields derived fields that can contain a list of fields.

BITFIELD

```
<FIELD NAME=”BITFIELD” >
     <CONTAINS NAME=”FIELD” />
</FIELD>
```

UNION

```
<FIELD NAME=”UNION” >
     <CONTAINS NAME=”FIELD” />
</FIELD>
```

RECORD

```
<FIELD NAME=”RECORD” >
     <CONTAINS NAME=”FIELD” />
</FIELD>
```

ARRAY

An array is a sequence of defined records. It can be presented to the user in a sequence or in a table.

Example:

```
<FIELD NAME=”ARRAY” >
     <PROPERTY NAME=”DIM” TYPE=”int” />

     <CONTAINS NAME=”RECORD” />
</FIELD>
```

6.9 Special Containers

We can now define interesting objects such as Command objects used for spacecraft commanding.

CMDS

```
<OBJECT NAME=”CMDS”>
     <PROPERTY NAME=”ACCESS” TYPE=”enum”
     VALUES=”NORMAL HAZARDOUS CRITICAL”/>
     <PROPERTY NAME=”ABSTRACT” TYPE=”boolean” />

     <CONTAINS NAME=”FIELD”/>
</OBJECT>
```

PACKET

SCRIPT

MESSAGE

6.10 CONTAINER CONTAINERS

SECTION

We can define a section as a container that can contain those interesting objects.

Example:

```
<OBJECT NAME=”SECTION” >
          <CONTAINS NAME=”CMDS”>
          …
</OBJECT>
```

DATABASE

A database now can be defined as a collection of sections.

Example:

```
<OBJECT NAME=”DATABASE” >
     <PROPERTY NAME=”VERSION” TYPE=”string” />

          <CONTAINS NAME=”SECTION”/>
</OBJECT>
```

7 Tools

Microsoft XML Parser
http://msdn.microsoft.com/downloads/tools/xmlparser/xmlparser.asp

Datachannel JAVA Parser http://xdev.datachannel.com/

SAX Parser in java http://www.megginson.com/SAX/index.html

Other tool list http://www.xmlephant.com/pages/Tools/Parsers/index.html

http://www.computer.org/internet/v2n3/xml-resource.htm

8 Reference Documents

External Documents

XML specification
http://www.w3.org/TR/1998/REC-xml-19980210

XSL specification
http://www.w3.org/TR/1998/WD-xsl-19981216.html

XML Namespace specification
http://www.w3.org/TR/1999/REC-xml-names-19990114/

XML-Schema specification
http://www.w3.org/TR/NOTE-xml-schema-req
http://www.w3.org/TR/xmlschema-1/
http://www.w3.org/TR/xmlschema-2/

Document Object Model
http://www.w3.org/TR/REC-DOM-Level-1/

8.1 Glossary

TBD

Appendices

Appendix A. Index of reserved SML Tags

Tag Name

ALRT

ARRAY

BITFIELD

CHAIN

CHAR

CMDS

DATA

DATABASE

DEFINE

DFP

DRANGE

FROM

HEADER

IFDEF

IFNDEF

INCLUDE

INFO

LOGFILE

MESSAGE

METHOD

NOTIFY

PACKET

RECORD

REPLY

REQUEST

ROUTE

SB

SCHEMA

SCRIPT

SECTION

SFP

SI

SLI

SML

STRING

TIME

TO

TRAILER

UB

UI

ULI

UNION

WARN

8.2 Appendix B. Command Database Example

```
<?xml version="1.0" standalone="no" ?>

<DATABASE NAME="FUSE_CMDS">

    <SCHEMA NAME="CCSDS" DESC="Protocol definition" >

        <CMDS NAME="CCSDS_CMD" DESC="CCSDS Root Command" >
           <HEADER MAXSIZE="240" DESC="CCSDS Header">
              <RECORD NAME="HDR1" >
                 <BITFIELD NAME="PKT">
                    <UI NAME="Version" MASK="0xE000" DEFAULT="0" MODE="STATIC" />
```

```
                    <UI NAME="Type" MASK="0x1000" DEFAULT="1" MODE="STATIC" />
                    <UI NAME="SecHdrFlag" MASK="0x800" DEFAULT="1" MODE="STATIC" />
                    <UI NAME="APID" MASK="0x7F" />
                </BITFIELD>
                <BITFIELD NAME="SEQ" >
                    <UI NAME="Flags" MASK="0xC000" DEFAULT="3" MODE="STATIC" />
                    <UI NAME="Count" MASK="0x3FF" DEFAULT="0" >
                        <METHOD NAME="CCSDS_SEQCOUNT" />
                    </UI>
                </BITFIELD>
                <UI NAME="PacketLength" MODE="STATIC" >
                    <METHOD NAME="CCSDS_PLEN" />
                </UI>
            </RECORD >
            <RECORD NAME="HDR2" >
                <UB NAME="SEDS_RESERVED" DEFAULT="0" MODE="STATIC" />
                <UB NAME="FCTN" DEFAULT="0" />
                <UB NAME="CHECKSUM" >
                    <METHOD NAME="CCSDS_CHECKSUM" />
                </UB>
            </RECORD>
        </HEADER>

        <!-- This is for testing only since the CCSDS checksum is in the secondary
header -->
        <TRAILER>
            <UB NAME="Checksum" >
                <METHOD NAME="CCSDS_TRAILER" />
            </UB>
        </TRAILER>
    </CMDS>


    <CCSDS_CMD NAME="IDS" DESC="Instrument Data System" ABSTRACT="TRUE" >
    </CCSDS_CMD>

    <IDS NAME="SWB" DESC="Software Bus Headers" ABSTRACT="TRUE" >
        <UB NAME="USER_DATA">0</UB>
        <UB NAME="OPCODE" >0 </UB>
    </IDS>

    <SWB NAME="FES_CMDS" DESC="Fine Error Sensor Commands" ABSTRACT="TRUE">
        <HEADER>
            <HDR1>
                <PKT>
                    <APID>111</APID>
                </PKT>
            </HDR1>
        </HEADER>

        <UNION NAME="FES" AS="FCTN">
            <UB NAME="FESA" DEFAULT="40" />
            <UB NAME="FESB" DEFAULT="46" />
        </UNION>
    </SWB>

    <SWB NAME="IDS_CMDS" ABSTRACT="TRUE">
        <!-- Other variant of syntax (Element names are unique) -->
        <APID>112</APID>
    </SWB>

    <IDS_CMDS NAME="SYSTEM0"
        DESC="IDS DATA I/O TASK (BOOT/NORMAL COMMON COMMANDS)" ABSTRACT="TRUE" >

    </IDS_CMDS>

    <IDS_CMDS NAME="BC_MGR"
            DESC="IDS IDB BUS CONTROLLER MANAGER TASK" ABSTRACT="TRUE">
        <HEADER>
            <HDR2>
                <FCTN>35</FCTN>
```

```
            </HDR2>
         </HEADER>
      </IDS_CMDS>

      <SYSTEM0 NAME="IDS_PPS"
            DESC="IDS 1HZ SIGNAL SOURCE SELECT (BOOT/NORMAL MODE" ABSTRACT="TRUE" >
         <OPCODE>21</OPCODE>
      </SYSTEM0>

      <BC_MGR NAME="IDS_IDBCTRL"
            DESC="START/STOP IDB OR SET RETRY PARAMETERS" >
         <USER_DATA>0</USER_DATA>
         <OPCODE>17</OPCODE>
         <BITFIELD NAME="PARAMETERS" >
            <UI NAME="OP" DEFAULT="0" MASK= "0x0003"
                  DESC="0 = START 1 = STOP 2 = ALTER RETRIES" />
            <UI NAME="PRIMARY_CHAN" DEFAULT="0" MASK="0x0004"
               DESC="Primary channel 0=A, 1=B" />
            <UI NAME="RETRY_CHAN" DEFAULT="0" MASK="0x0008"
                  DESC="Retry channel 0=SAME, 1=ALTERNATE" />
            <UI NAME="UNUSED" DEFAULT="0" MASK="0xFFF0" />
         </BITFIELD>
      </BC_MGR>

      <IDS_IDBCTRL NAME="IDSIDBDIS" >
         <PARAMETERS>
            <OP>1</OP>
         </PARAMETERS>
         <MODIFY NAME="USER_DATA" MIN="12" MAX="14" />
      </IDS_IDBCTRL>

      <IDS_IDBCTRL NAME="IDSIDBENA" >
         <PARAMETERS>
            <OP>0</OP>
         </PARAMETERS>
      </IDS_IDBCTRL>

      <FES_CMDS NAME="IFES_FULLIMG" DESC="CONFIGURE THE FULL FIELD OF VIEW IMAGE" >
         <USER_DATA>0</USER_DATA>
         <OPCODE>17</OPCODE>

         <UB NAME="HW_BIN_X">0</UB>
         <UB NAME="HW_BIN_Y">0</UB>
      </FES_CMDS>


      <FES_CMDS NAME="IFES_BADPIX" DESC="SET BAD PIXEL" >
         <USER_DATA>0</USER_DATA>
         <OPCODE>25</OPCODE>

         <UB NAME="NUM_BAD_PIXELS">0</UB>
         <UB NAME="RESERVED4">0</UB>
         <UI NAME="INTENSITY">0</UI>

         <RECORD NAME="BP" SIZE="39" DESC="Bad Pixel Array" >
            <UI NAME="X">0</UI>
            <UI NAME="Y">0</UI>
            <UI NAME="VALUE">0</UI>
         </RECORD>
      </FES_CMDS>
   </SCHEMA>
</DATABASE>
```

8.3 Appendix C. Telemetry Packet Definition Example

```
<?xml version="1.0" standalone="no" ?>

<DATABASE NAME="FUSE_TLM" >

     <SECTION NAME="FUSE IDS Telemetry" >
        <PACKET NAME="CCSDS_PKT_HDR" DESC="CCSDS Header" >
           <BITFIELD NAME="HDR1" DESC="CCSDS Header 1st 16 bits" >
              <UI NAME="PVNO" MASK="0xE0000"
                    DESC="Packet Version Number  Bits 0-2" />
              <UI NAME="PCKT" MASK="0x1000"
                    DESC="Packet Type            Bit    3" />
              <UI NAME="SHDF" MASK="0x100"
                    DESC="Secondary Header Flag  Bit    4" />
              <UI NAME="APID" MASK="0x7FF"
                 DESC="Application ID         Bits 5-15" />
           </BITFIELD>

           <BITFIELD NAME="HDR2" DESC="CCSDS Header 2nd 16 bits" >
              <UI NAME="SEGF"   MASK="0xC000" DESC="Segment Flags          Bits  0-1"
/>
              <UI NAME="SCNT"   MASK="0x0x3000" DESC="Source Sequence Count  Bits  2-
15" />
           </BITFIELD>

           <UI NAME="PLEN" DESC="Packet Length" />
           <ULI  NAME="STIME_SECONDS" />
           <ULI  NAME="STIME_SUBSECONDS" />

           <BITFIELD NAME="HDR3"      DESC="Function Code 8 bits" >
              <UB NAME="SPARE" MASK="0x80" DESC="Spare Bit 0" />
              <UB NAME="FCTN"   MASK="0x7F" DESC="Function Code" />
           </BITFIELD>

           <UB NAME="CHKSUM"    DESC="Checksum" />
        </PACKET>

        <CCSDS_PACKET NAME="IDS_PACKET">
           <UNION NAME="USRDATA" >


              <UB NAME="USRDATA"
                 DESC="User Data" TYPE="US64" MASK="0xFF" UNITS="counts" />

              <!-- Detector PHA good/bad flag, accumulation period: used only in
packets I_D1_PHASEG1, I_D1_PHASEG2, I_D2_PHASEG1, I_D2_PHASEG2 -->
              <UB NAME="ACC_PERIOD"
                 DESC="Detector PHA accumulation period: used only in packets
I_D1_PHASEG1, I_D1_PHASEG2, I_D2_PHASEG1, I_D2_PHASEG2"
                 TYPE="RTDS" MASK="0x02" UNITS="counts" >
                 <DRANGE>
                    <OPTION VALUE="0" NAME="8 SECONDS" />
                    <OPTION VALUE="1" NAME="128 SECONDS" />
                 </DRANGE>
              </UB>

              <UB NAME="PHA_GOOD"
                 DESC="Detector PHA status: (0=good, 1=bad) used only in packets
I_D1_PHASEG1, I_D1_PHASEG2, I_D2_PHASEG1, I_D2_PHASEG2"
                 TYPE="RTDS" MASK="0x01" UNITS="counts" >
                 <DRANGE>
                    <OPTION VALUE="0" NAME="GOOD" />
                    <OPTION VALUE="1" NAME="BAD" />
                 </DRANGE>
              </UB>

              <!-- Subsystem side A/B, 1/2 good/bad flags: used only in packets
I_PSDU_HSK, I_MAE_HSK, I_FES_HSK, I_DET_T26_HSK, I_DET_T27_HSK, I_DET_T27_HSK,
I_FPA_HSK -->

              <UB NAME="B_OR_2_GOOD"
                 DESC="Subsystem B or 2 status: (0=good, 1=bad) used only in packets
```

```
I_PSDU_HSK, I_MAE_HSK, I_FES_HSK, I_DET_T26_HSK, I_DET_T27_HSK, I_DET_T27_HSK,
I_FPA_HSK"
                  TYPE="RTDS" MASK="0x02" UNITS="counts" >
                  <DRANGE>
                     <OPTION VALUE="0" NAME="GOOD" />
                     <OPTION VALUE="1" NAME="BAD" />
                  </DRANGE>
               </UB>

               <UB NAME="A_OR_1_GOOD"
                  DESC="Subsystem A or 1 status: (0=good, 1=bad) used only in packets
I_PSDU_HSK, I_MAE_HSK, I_FES_HSK, I_DET_T26_HSK, I_DET_T27_HSK, I_DET_T27_HSK,
I_FPA_HSK"
                  TYPE="RTDS" MASK="0x01" UNITS="counts" >
                  <DRANGE>
                     <OPTION VALUE="0" NAME="GOOD" />
                     <OPTION VALUE="1" NAME="BAD" />
                  </DRANGE>
               </UB>

            </UNION>
            <UB NAME="OPCODE"    DESC="IDS Opcode" />
         </CCSDS_PACKET>
      </SECTION>

      <SECTION NAME="IDS BOOT" >
         <IDS_PACKET NAME="IDS_MDUMP"
                 DESC = "FUSE IDS TELEMETRY MEMORY DUMP FIRST/REMAINING PACKET" >

            <APID>63</APID>
            <FCTN>1</FCTN>
            <OPCODE>55</OPCODE>

            <ULI NAME="ADDRESS" TYPE="RTAS" />
            <ULI NAME="REMAIN" TYPE="RTAS" />
            <UB  NAME="DATA" SIZE="997" />
         </IDS_PACKET>

         <IDS_PACKET NAME="ODUMP"
                DESC = "FUSE IDS TELEMETRY OBJECT DUMP FIRST/REMAINING PACKET" >
            <APID>63</APID>

            <UI  NAME="OBJECT_ID" />
            <UI  NAME="P_REMAIN" DESC="PACKETS REMAINING EXCLUDING THIS ONE" />
            <ULI NAME="T_REMAIN" DESC="TOTAL REMAING 16 BIT WORDS" />
            <ULI NAME="ADDRESS" />
            <UB  NAME="DATA" SIZE="994" />
         </IDS_PACKET>

         <!-- IDS BOOT HOUSEKEEPING TELEMETRY -->


         <IDS_PACKET NAME="BOOT_HSK"
                 DESC = "FUSE IDS HOUSEKEEPING TELEMETRY" >

            <APID>63</APID>
            <FCTN>1</FCTN>

         <!-- This part of the IDS housekeeping comes primarily from the boot stage, I
call system -->

            <UNION NAME="SYS_STATUS1" >
               <UB NAME="MODE"     MASK="0x80" />
               <UB NAME="RESTART"  MASK="0x40" />
               <UB NAME="BANK"     MASK="0x20" />
               <UB NAME="PORTMODE" MASK="0x18" />
               <UB NAME="CLOCKSRC" MASK="0x04" />
               <UB NAME="PWRDC"    MASK="0x02" />
               <UB NAME="PWRAC"    MASK="0x01" />
            </UNION>
```

```
            <BITFIELD NAME="SYS_STATUS2" >
               <UB NAME="MDUMPPROG"          MASK="0x80" />
               <UB NAME="MCOPYPROG"          MASK="0x40" />
               <UB NAME="BITPROG"                  MASK="0x20" />
               <UB NAME="APPLBOOTPROG"       MASK="0x10" />
               <UB NAME="CLOCKABSENT"        MASK="0x08" />
               <UB NAME="RESERVED1"          MASK="0x07" />
            </BITFIELD>

            <ULI NAME="COUNTER" />
            <UI  NAME="NUM_EXE_CMDS" />
            <UI  NAME="NUM_REJ_CMDS" />
            <UB  NAME="LAST_CMD_EXE" />
            <UB  NAME="LAST_CMD_REJ" />


            <BITFIELD NAME="LAST_CMD_EXE_TIME" >
               <ULI  NAME="TIME_LAST_CMD" MASK="0xFFFFFF00" DESC="24 Bits of MET" />
               <ULI  NAME="SUBSECONDS"    MASK="0x000000FF" DESC="8 BITS OF SUBSECONDS"
/>
            </BITFIELD>

            <UI NAME="LAST_CMD_REJ_REASON" />

            <UI NAME="BAD_TCU_COUNT" />

            <!-- LAST 6 WORDS OF CMD PACKET -->

            <UI NAME="CMD_WORD1" />
            <UI NAME="CMD_WORD2" />
            <UI NAME="CMD_WORD3" />
            <UI NAME="CMD_WORD4" />
            <UI NAME="CMD_WORD5" />
            <UI NAME="CMD_WORD6" />

            <!-- LAST 6 WORDS OF DUMP PACKET -->
            <UI NAME="DUMP_WORD1" />
            <UI NAME="DUMP_WORD2" />
            <UI NAME="DUMP_WORD3" />
            <UI NAME="DUMP_WORD4" />
            <UI NAME="DUMP_WORD5" />
            <UI NAME="DUMP_WORD6" />

            <UI NAME="SELFTEST_RESULTS" />

            <ULI NAME="INTERRUPTS" />
            <ULI NAME="TCU" />

            <UI NAME="DEBUG1" />
            <UI NAME="DEBUG2" />
            <UI NAME="DEBUG3" />
         </IDS_PACKET>

      </SECTION>

      <SECTION NAME="HOUSEKEEPING TELEMETRY" >
         <IDS_PACKET NAME="IDS_HOUS"
            DESC = "FUSE IDS HOUSEKEEPING TELEMETRY" >
            <APID>64</APID>
            <FCTN>01</FCTN>

            <UNION NAME="OPTIONS1">
               <UB NAME="MODE"     MASK="1xb10000000" />
               <UB NAME="RESTART"  MASK="1x01000000" />
               <UB NAME="BANK"     MASK="1x00100000" />
               <UI NAME="PORTMODE" MASK="1x0001100011111111" />
               <UB NAME="DUMP"     MASK="1x00000100" />
               <UB NAME="COPY"     MASK="1x00000010" />
               <UB NAME="RESERV1"  MASK="1x01000001" />
            </UNION>
```

```
            <UI    NAME="RESERV2" />
            <ULI   NAME="COUNTER" />
            <UI    NAME="RX_CMDS" />
            <UI    NAME="EX_CMDS" />
            <UI    NAME="LAST_CMD" />
            <UI    NAME="LCMD_NOT" />

        </IDS_PACKET>
     </SECTION>
</DATABASE>
```

8.4 Appendix D. Message Definition Example

```
<?xml version="1.0" standalone="no" ?>

<DATABASE NAME="SCL_MESSAGES" >
     <SECTION NAME="SCL Compiler Messages" >
        <SECTION NAME="Input Messages" >
           <MESSAGE NAME="scl_do" >
               <REQUEST>
                  <STRING NAME="cmd" DESC="SCL ASCII command string" />
                  <UI NAME="user" DESC="User id" />
               </REQUEST>

               <REPLY>
                  <SI NAME="status" />
                  <STRING NAME="msg" required="no" />
                  </REPLY>
            </MESSAGE>

            <MESSAGE NAME="scl_get_file_from_id"
                   DESC="find the file name from script id" >
               <REQUEST>
                  <UI NAME="blockid" DESC="script or rule id" />
               </REQUEST>
               <REPLY>
                  <SI NAME="status" />
                  <STRING NAME="filename" required="no" DESC="File name" />
               </REPLY>
            </MESSAGE>

            <MESSAGE NAME="scl_count_types"
               DESC="Returns the number of object types defined in current project" >

               <REQUEST />
               <REPLY >
                  <ULI NAME="count"
                      DESC="Number of object types defined in current project" />
               </REPLY>
            </MESSAGE>

            <MESSAGE NAME="scl_get_index_type" >
               <REQUEST>
                  <ULI NAME="index" DESC="Nth element" />
               </REQUEST>
               <REPLY>
                  <ULI NAME="type" />
               </REPLY>
            </MESSAGE>

           <MESSAGE NAME="scl_count_resources"
                   DESC="Returns the number of objects of a particular type" >
               <REQUEST>
                  <ULI NAME="type" DESC="Resource type" />
               </REQUEST>
```

```
            <REPLY>
               <ULI NAME="count" DESC="count of resources of that type" />
            </REPLY>
         </MESSAGE>

      </SECTION>
      <SECTION NAME="Output Messages" >
      </SECTION>
   </SECTION>
   <SECTION NAME="RTE Messages" >
      <SECTION NAME="Output Messages" >
      </SECTION>
   </SECTION>
</DATABASE>
```

8.5 Appendix E. Database Schema Example

```
<?xml version="1.0" standalone="no" ?>

<DATABASE NAME="Database Templates" >

    <SCHEMA NAME="FUSE Database Schema">

        <RECORD NAME="DBIT" DESC="DBItem" ID="5000">
            <UI NAME="recType" DEFAULT="0"/>

            <BITFIELD NAME="flags" >
                <UI NAME="trace" MASK="0x1" />
                <UI NAME="IEprocessing" MASK="0x2" />
                <UI NAME="known" MASK="0x4" />
                <UI NAME="determined" MASK="0x8" />
                <UI NAME="unused01" MASK="0x10" />
                <UI NAME="unused02" MASK="0x20" />
                <UI NAME="performDeltaLimitChecking" MASK="0x40" />
                <UI NAME="performRailLimitChecking" MASK="0x80" />
                <UI NAME="rejectSource" MASK="0x100" />
                <UI NAME="performDataioArchiving" MASK="0x200" />
                <UI NAME="inhibitChangeDetect" MASK="0x400" />
                <UI NAME="inhibitEUConversion" MASK="0x800" />
                <UI NAME="inhibitRTCO" MASK="0x1000" />
                <UI NAME="performAlarmChecking" MASK="0x2000" />
                <UI NAME="performSmoothing" MASK="0x4000" />
                <UI NAME="unused05" MASK="0x8000" />
            </BITFIELD>
        </RECORD>

        <DBIT NAME="SENS" DESC="Sensor" ID="5001" >
            <BITFIELD NAME="userFlags" >
                <UI NAME="beforeProcessing" MASK="0x1" />
                <UI NAME="beforeEUConversion" MASK="0x2" />
                <UI NAME="beforeSmoothing" MASK="0x4" />
                <UI NAME="beforeAlarmChecking" MASK="0x8" />
                <UI NAME="afterProcessing" MASK="0x10" />
                <UI NAME="reserved01" MASK="0x20" />
                <UI NAME="reserved02" MASK="0x40" />
                <UI NAME="reserved03" MASK="0x80" />
                <UI NAME="userDefined01" MASK="0x100" />
                <UI NAME="userDefined02" MASK="0x200" />
                <UI NAME="userDefined03" MASK="0x400" />
                <UI NAME="userDefined04" MASK="0x800" />
                <UI NAME="userDefined05" MASK="0x1000" />
                <UI NAME="userDefined06" MASK="0x2000" />
                <UI NAME="userDefined07" MASK="0x4000" />
                <UI NAME="userDefined08" MASK="0x8000" />
            </BITFIELD>
            <SI NAME="alarmLevel" DEFAULT="0" />
            <SLI NAME="rawValue" DEFAULT="0" />
            <SFP NAME="deltaLimit" DEFAULT="0.0" />
```

```
    <UI NAME="staleCounter" DEFAULT="0" />
    <UI NAME="stalePeriod" DEFAULT="60" />
</DBIT>


<DBIT NAME="DERV" DESC="Derived_Item" ID="5003">
    <BITFIELD NAME="userFlags" >
        <UI NAME="beforeProcessing" MASK="0x1" />
        <UI NAME="beforeEUConversion" MASK="0x2" />
        <UI NAME="beforeSmoothing" MASK="0x4" />
        <UI NAME="beforeAlarmChecking" MASK="0x8" />
        <UI NAME="afterProcessing" MASK="0x10" />
        <UI NAME="reserved01" MASK="0x20" />
        <UI NAME="reserved02" MASK="0x40" />
        <UI NAME="reserved03" MASK="0x80" />
        <UI NAME="userDefined01" MASK="0x100" />
        <UI NAME="userDefined02" MASK="0x200" />
        <UI NAME="userDefined03" MASK="0x400" />
        <UI NAME="userDefined04" MASK="0x800" />
        <UI NAME="userDefined05" MASK="0x1000" />
        <UI NAME="userDefined06" MASK="0x2000" />
        <UI NAME="userDefined07" MASK="0x4000" />
        <UI NAME="userDefined08" MASK="0x8000" />
    </BITFIELD>
    <SI NAME="alarmLevel" DEFAULT="0" />
</DBIT>

<DBIT NAME="USEN" DESC="Unsigned_Sensor" ID="5004">
    <BITFIELD NAME="userFlags">
        <UI NAME="beforeProcessing" MASK="0x1" />
        <UI NAME="beforeEUConversion" MASK="0x2" />
        <UI NAME="beforeSmoothing" MASK="0x4" />
        <UI NAME="beforeAlarmChecking" MASK="0x8" />
        <UI NAME="afterProcessing" MASK="0x10" />
        <UI NAME="reserved01" MASK="0x20" />
        <UI NAME="reserved02" MASK="0x40" />
        <UI NAME="reserved03" MASK="0x80" />
        <UI NAME="userDefined01" MASK="0x100" />
        <UI NAME="userDefined02" MASK="0x200" />
        <UI NAME="userDefined03" MASK="0x400" />
        <UI NAME="userDefined04" MASK="0x800" />
        <UI NAME="userDefined05" MASK="0x1000" />
        <UI NAME="userDefined06" MASK="0x2000" />
        <UI NAME="userDefined07" MASK="0x4000" />
        <UI NAME="userDefined08" MASK="0x8000" />
    </BITFIELD>
    <SI NAME="alarmLevel" DEFAULT="0" />
    <ULI NAME="rawValue" DEFAULT="0" />
    <SFP NAME="deltaLimit" DEFAULT="0.0" />
    <UI NAME="staleCounter" DEFAULT="0" />
    <UI NAME="stalePeriod" DEFAULT="60" />
</DBIT>

-------------------------------------------------------

<DBIT NAME="SCMD" DESC="Actuator" ID="0">
    <recType>0</recType>

    <UI NAME="rawValue" DEFAULT="0" />
    <UI NAME="access" DEFAULT="0" />
</DBIT>

<SENS NAME="RTAS" DESC="Analog_Sensor" ID="1" >
    <recType>1</recType>

    <SFP NAME="engValue" DEFAULT="0.0" />
    <SFP NAME="smoothedValue" DEFAULT="0.0" />
    <SFP NAME="smoothingFactor" DEFAULT="0.0" />
    <SFP NAME="sensitivity" DEFAULT="0.0" />
    <SFP NAME="highRed" DEFAULT="0.0" />
    <SFP NAME="highYellow" DEFAULT="0.0" />
```

```
    <SFP NAME="lowYellow" DEFAULT="0.0" />
    <SFP NAME="lowRed" DEFAULT="0.0" />
    <UI NAME="coeffID" DEFAULT="0" />
</SENS>

<SENS NAME="RTDS" DESC="Discrete_Sensor" ID="2" >
    <recType>2</recType>

    <UI NAME="stateID" DEFAULT="0" />
    <UI NAME="persistLevel" DEFAULT="1" />
    <UI NAME="persistCount" DEFAULT="0" />
</SENS>


<DBIT NAME="COFM" DESC="Coefficient_Family" ID="3" >
    <recType>3</recType>

    <UI NAME="fill" />

    <UI NAME="numOfCoeffs" />
    <ARRAY NAME="Coeffs" LENGTH="numOfCoeffs">
       <SFP NAME="Coef"/>
    </ARRAY>
</DBIT>


<DBIT NAME="STFM" DESC="State_Family" ID="5" >
    <recType>5</recType>

    <STRING NAME="states" />
    <STRING NAME="colors" />
</DBIT>

<DERV NAME="RTAD" DESC="Analog_Derived" ID="6" >
    <recType>6</recType>

    <SFP NAME="value" DEFAULT="0.0" />
    <SFP NAME="sensitivity" DEFAULT="0.0" />
    <SFP NAME="highRed" DEFAULT="0.0" />
    <SFP NAME="highYellow" DEFAULT="0.0" />
    <SFP NAME="lowYellow" DEFAULT="0.0" />
    <SFP NAME="lowRed" DEFAULT="0.0" />
</DERV>


<DERV NAME="RTDD" DESC="Discrete_Derived" ID="7">
    <recType>7</recType>
    <SLI NAME="value" DEFAULT="0" />
    <UI NAME="stateID" DEFAULT="0" />
</DERV>

<SCMD NAME="RTAA" DESC="Abstract_Actuator" ID="8">
    <recType>8</recType>
    <UI NAME="onID" />
    <UI NAME="offID" />
    <SI NAME="value" />
    <SI NAME="side" />
</SCMD>

<SCMD NAME="RTDA" DESC="DiscreteActuator" ID="9" >
    <recType>9</recType>
    <UI NAME="stateID" DEFAULT="0" />
</SCMD>

<DBIT NAME="RTTM" DESC="Time_Actuator" ID="10" >
    <recType>10</recType>
    <SLI NAME="time" />
</DBIT>

<DBIT NAME="RTVR" DESC="DB_Version" ID="11" >
    <recType>11</recType>
```

```
            <SFP NAME="version" />
         </DBIT>

         <DBIT NAME="BINA" DESC="Binary_Loads" ID="12" >
            <recType>12</recType>

            <UI NAME="length" />

            <ARRAY NAME="values" LENGTH="length" >
               <UB NAME="value"/>
            </ARRAY>

         </DBIT>

         <RTDA NAME="CEXL" DESC="CEXL_Commands" ID="13">
            <recType>13</recType>
         </RTDA>


         <DBIT NAME="DCOM" DESC="Decom_Standard" ID="15">
            <recType>15</recType>

            <UI NAME="numRecords" />
            <ARRAY NAME="records" LENGTH="numRecords" >
               <RECORD NAME="item">
                  <UI NAME="recordID" />
                  <UI NAME="startBit" />
                  <UI NAME="numBits" />
               </RECORD>
            </ARRAY>
         </DBIT>


         <DBIT NAME="DCMC" DESC="Decom_CCSDS" ID="16">
            <recType>16</recType>

            <ULI NAME="numRecords" />
            <ARRAY NAME="records" LENGTH="numRecords" >
               <RECORD NAME="item" >
                  <UI NAME="recordID" />
                  <UI NAME="dataStructure" />
                  <UI NAME="pointerIncrement" />
                  <UI NAME="signExtend" />
                  <ULI NAME="mask" />
                  <UI NAME="lshift" />
                  <UI NAME="order" />
               </RECORD>
            </ARRAY>
         </DBIT>

         <DBIT NAME="DIMG" DESC="Decom_Image" ID="17">
            <recType>17</recType>

            <ULI NAME="offset" DEFAULT="0" DESC="offset to memory image data from start
of packet" />
            <ULI NAME="length" DEFAULT="0" DESC="length of memory image data" />
            <ULI NAME="handlerID" DEFAULT="0" DESC="handler for App Specific image
handler" />
         </DBIT>

         <DBIT NAME="QUAT" DESC="Quaternion_Record" ID="28">
            <recType>28</recType>

            <DFP NAME="Q1" DEFAULT="0.0" />
            <DFP NAME="Q2" DEFAULT="0.0" />
            <DFP NAME="Q3" DEFAULT="0.0" />
            <DFP NAME="Q4" DEFAULT="0.0" />
         </DBIT>

         <SENS NAME="AS64" DESC="Analog_Sensor_64_bit" ID="29">
            <recType>29</recType>
```

```
    <ULI NAME="filler" />
    <DFP NAME="engValue" DEFAULT="0.0" />
    <DFP NAME="smoothedValue" DEFAULT="0.0" />
    <DFP NAME="smoothingFactor" DEFAULT="0.0" />
    <DFP NAME="sensitivity" DEFAULT="0.0" />
    <DFP NAME="highRed" DEFAULT="0.0" />
    <DFP NAME="highYellow" DEFAULT="0.0" />
    <DFP NAME="lowYellow" DEFAULT="0.0" />
    <DFP NAME="lowRed" DEFAULT="0.0" />
    <UI NAME="coeffID" DEFAULT="0" />
</SENS>

<DERV NAME="AD64" DESC="Analog_Derived_64" ID="30">
    <recType>30</recType>

    <DFP NAME="value" DEFAULT="0.0" />
    <DFP NAME="sensitivity" DEFAULT="0.0" />
    <DFP NAME="highRed" DEFAULT="0.0" />
    <DFP NAME="highYellow" DEFAULT="0.0" />
    <DFP NAME="lowYellow" DEFAULT="0.0" />
    <DFP NAME="lowRed" DEFAULT="0.0" />
</DERV>

<DBIT NAME="PSTR" DESC="PString" ID="31">
    <recType>31</recType>

    <BITFIELD NAME="userFlags" >
        <UI NAME="beforeProcessing" MASK="0x1" />
        <UI NAME="beforeEUConversion" MASK="0x2" />
        <UI NAME="beforeSmoothing" MASK="0x4" />
        <UI NAME="beforeAlarmChecking" MASK="0x8" />
        <UI NAME="afterProcessing" MASK="0x10" />
        <UI NAME="reserved01" MASK="0x20" />
        <UI NAME="reserved02" MASK="0x40" />
        <UI NAME="reserved03" MASK="0x80" />
        <UI NAME="userDefined01" MASK="0x100" />
        <UI NAME="userDefined02" MASK="0x200" />
        <UI NAME="userDefined03" MASK="0x400" />
        <UI NAME="userDefined04" MASK="0x800" />
        <UI NAME="userDefined05" MASK="0x1000" />
        <UI NAME="userDefined06" MASK="0x2000" />
        <UI NAME="userDefined07" MASK="0x4000" />
        <UI NAME="userDefined08" MASK="0x8000" />
    </BITFIELD>

    <UI NAME="maxlength" DEFAULT="512"
        DESC="max chars really one less due to null at end" />

    <UI NAME="length" DEFAULT="0" />
    <ARRAY NAME="cstr" LENGTH="length" >
        <UB NAME="str" DESC="note this is null terminated string" />
    </ARRAY>

</DBIT>

<USEN NAME="ULAS" DESC="Unsigned_Analog_Sensor" ID="32" >
    <recType>32</recType>
    <SFP NAME="engValue" DEFAULT="0.0" />
    <SFP NAME="smoothedValue" DEFAULT="0.0" />
    <SFP NAME="smoothingFactor" DEFAULT="0.0" />
    <SFP NAME="sensitivity" DEFAULT="0.0" />
    <SFP NAME="highRed" DEFAULT="0.0" />
    <SFP NAME="highYellow" DEFAULT="0.0" />
    <SFP NAME="lowYellow" DEFAULT="0.0" />
    <SFP NAME="lowRed" DEFAULT="0.0" />
    <UI NAME="coeffID" DEFAULT="0" />
</USEN>

<USEN NAME="ULDS" DESC="Unsigned_Discrete_Sensor" ID="33">
    <recType>33</recType>
```

```
            <UI NAME="stateID" />
            <UI NAME="persistLevel" DEFAULT="1" />
            <UI NAME="persistCount" DEFAULT="0" />
         </USEN>

         <USEN NAME="US64" DESC="Unsigned_Analog_Sensor_64_bit" ID="34">
            <recType>34</recType>

            <ULI NAME="filler" />
            <DFP NAME="engValue" DEFAULT="0.0" />
            <DFP NAME="smoothedValue" DEFAULT="0.0" />
            <DFP NAME="smoothingFactor" DEFAULT="0.0" />
            <DFP NAME="sensitivity" DEFAULT="0.0" />
            <DFP NAME="highRed" DEFAULT="0.0" />
            <DFP NAME="highYellow" DEFAULT="0.0" />
            <DFP NAME="lowYellow" DEFAULT="0.0" />
            <DFP NAME="lowRed" DEFAULT="0.0" />
            <UI NAME="coeffID" DEFAULT="0" />
         </USEN>
   </SCHEMA>
</DATABASE>
```

8.6 Appendix F. Database Records Example

```
<?xml version="1.0" standalone="no" ?>

<!-- Interface & Control Systems -->
<!-- SML (c) 1999 - All Rights Reserved -->
<!-- Fuse Telemetry Example -->
<!-- Pat Cappelaere Oct 1999 -->

<DATABASE NAME="FUSE Telemetry Database" >
     <SECTION NAME="Decoms" >
         <DCMC NAME="MDUMP" ID="1">
            <flags>0</flags>

            <ARRAY NAME="item" INDEX="numRecords" LENGTH="2">
               <RECORD>
                  <recordID>400</recordID>
                  <dataStructure>4</dataStructure>
                  <pointerIncrement>0</pointerIncrement>
                  <signExtend>0</signExtend>
                  <mask>4294967295</mask>
                  <lshift>0</lshift>
                  <order>0</order>
               </RECORD>

               <RECORD>
                  <recordID>401</recordID>
                  <dataStructure>4</dataStructure>
                  <pointerIncrement>4</pointerIncrement>
                  <signExtend>0</signExtend>
                  <mask>4294967295</mask>
                  <lshift>0</lshift>
                  <order>0</order>
               </RECORD>
            </ARRAY>
         </DCMC>

         <DCMC NAME="IDS_HOUS" ID="2" >
            <flags>0</flags>

            <ARRAY NAME="item" INDEX="numRecords" LENGTH="13">
               <RECORD>
                  <recordID>402</recordID>
                  <dataStructure>1</dataStructure>
                  <pointerIncrement>0</pointerIncrement>
                  <signExtend>0</signExtend>
```

```
    <mask>128</mask>
    <lshift>-7</lshift>
    <order>0</order>
</RECORD>

<RECORD>
    <recordID>403</recordID>
    <dataStructure>1</dataStructure>
    <pointerIncrement>0</pointerIncrement>
    <signExtend>0</signExtend>
    <mask>64</mask>
    <lshift>-6</lshift>
    <order>0</order>
</RECORD>

<RECORD>
    <recordID>404</recordID>
    <dataStructure>1</dataStructure>
    <pointerIncrement>0</pointerIncrement>
    <signExtend>0</signExtend>
    <mask>32</mask>
    <lshift>-5</lshift>
    <order>0</order>
</RECORD>

<RECORD>
    <recordID>405</recordID>
    <dataStructure>2</dataStructure>
    <pointerIncrement>0</pointerIncrement>
    <signExtend>0</signExtend>
    <mask>6399</mask>
    <lshift>-3</lshift>
    <order>0</order>
</RECORD>

<RECORD>
    <recordID>406</recordID>
    <dataStructure>2</dataStructure>
    <pointerIncrement>0</pointerIncrement>
    <signExtend>0</signExtend>
    <mask>4</mask>
    <lshift>-2</lshift>
    <order>0</order>
</RECORD>

<RECORD>
    <recordID>407</recordID>
    <dataStructure>1</dataStructure>
    <pointerIncrement>0</pointerIncrement>
    <signExtend>0</signExtend>
    <mask>2</mask>
    <lshift>-1</lshift>
    <order>0</order>
</RECORD>

<RECORD>
    <recordID>408</recordID>
    <dataStructure>1</dataStructure>
    <pointerIncrement>0</pointerIncrement>
    <signExtend>0</signExtend>
    <mask>65</mask>
    <lshift>0</lshift>
    <order>0</order>
</RECORD>

<RECORD>
    <recordID>409</recordID>
    <dataStructure>2</dataStructure>
    <pointerIncrement>0</pointerIncrement>
    <signExtend>0</signExtend>
    <mask>65535</mask>
```

```
                <lshift>0</lshift>
                <order>0</order>
            </RECORD>

            <RECORD>
                <recordID>410</recordID>
                <dataStructure>4</dataStructure>
                <pointerIncrement>3</pointerIncrement>
                <signExtend>0</signExtend>
                <mask>4294967295</mask>
                <lshift>0</lshift>
                <order>0</order>
            </RECORD>

            <RECORD>
                <recordID>411</recordID>
                <dataStructure>2</dataStructure>
                <pointerIncrement>7</pointerIncrement>
                <signExtend>0</signExtend>
                <mask>65535</mask>
                <lshift>0</lshift>
                <order>0</order>
            </RECORD>

            <RECORD>
                <recordID>412</recordID>
                <dataStructure>2</dataStructure>
                <pointerIncrement>9</pointerIncrement>
                <signExtend>0</signExtend>
                <mask>65535</mask>
                <lshift>0</lshift>
                <order>0</order>
            </RECORD>

            <RECORD>
                <recordID>413</recordID>
                <dataStructure>2</dataStructure>
                <pointerIncrement>11</pointerIncrement>
                <signExtend>0</signExtend>
                <mask>65535</mask>
                <lshift>0</lshift>
                <order>0</order>
            </RECORD>

            <RECORD>
                <recordID>414</recordID>
                <dataStructure>2</dataStructure>
                <pointerIncrement>13</pointerIncrement>
                <signExtend>0</signExtend>
                <mask>65535</mask>
                <lshift>0</lshift>
                <order>0</order>
            </RECORD>
        </ARRAY>
    </DCMC>

</SECTION>

<SECTION NAME="RECORDS" >
    <COFM ID="399" NAME="Coefs" >
        <ARRAY NAME="Coefs" INDEX="numOfCoeffs" LENGTH="3">
            <RECORD>
                <Coef>1.1</Coef>
            </RECORD>
            <RECORD>
                <Coef>2.1</Coef>
            </RECORD>
            <RECORD>
                <Coef>3</Coef>
            </RECORD>
        </ARRAY>
```

```
</COFM>

<RTAS ID="400" NAME="MDUMP_ADDRESS">
   <rawValue>0</rawValue>
</RTAS>

<RTAS ID="401" NAME="MDUMP_REMAIN">
   <rawValue>0</rawValue>
</RTAS>

<RTDS ID="402" NAME="IDS_HOUS_MODE">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="403" NAME="IDS_HOUS_RESTART">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="404" NAME="IDS_HOUS_BANK">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="405" NAME="IDS_HOUS_PORTMODE">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="406" NAME="IDS_HOUS_DUMP">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="407" NAME="IDS_HOUS_COPY">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="408" NAME="IDS_HOUS_RESERV1">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="409" NAME="IDS_HOUS_RESERV2">
   <rawValue>0</rawValue>
</RTDS>

<RTAS ID="410" NAME="IDS_HOUS_COUNTER">
   <rawValue>0</rawValue>
</RTAS>

<RTDS ID="411" NAME="IDS_HOUS_RX_CMDS">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="412" NAME="IDS_HOUS_EX_CMDS">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="413" NAME="IDS_HOUS_LAST_CMD">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="414" NAME="IDS_HOUS_LCMD_NOT">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="415" NAME="IDS_BIT_RAM_A">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="416" NAME="IDS_BIT_RAM_B">
   <rawValue>0</rawValue>
</RTDS>

<RTDS ID="417" NAME="IDS_BIT_SDB_CHIP">
```

```
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="418" NAME="IDS_BIT_SDB_RAM">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="419" NAME="IDS_BIT_ACT_RAM">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="420" NAME="IDS_BIT_FPU">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="421" NAME="IDS_BIT_PROM">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="422" NAME="IDS_BIT_EEPROM_A">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="423" NAME="IDS_BIT_EEPROM_B">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="424" NAME="IDS_BIT_TIMER">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="425" NAME="IDS_BIT_IDB_CHIP">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="426" NAME="IDS_BIT_IDB_RAM">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="427" NAME="IDS_BIT_RESERVE1">
            <rawValue>0</rawValue>
        </RTDS>


        <RTDS ID="428" NAME="IDS_BIT_RESERVE2">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="429" NAME="IDS_BIT_RESERVE3">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="430" NAME="IDS_BIT_RESERVE4">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="431" NAME="IDS_BIT_RESERVE5">
            <rawValue>0</rawValue>
        </RTDS>

        <RTDS ID="432" NAME="IDS_BIT_RESERVE6">
            <rawValue>0</rawValue>
            <stateID>433</stateID>
        </RTDS>

        <STFM ID="433" NAME="IDS_BIT_RESERVE6_STATES">
            <states>"0 open 1 closed"</states>
        </STFM>

    </SECTION>
</DATABASE>
```

8.7 Appendix G. Logging Example

```
<?xml version="1.0" standalone="no" ?>

<LOGFILE NAME="log.xml" CREATED="12/12/12 12:12:12.00" >
     <INFO TIME="12/12/12 12:00:00.1" SOURCE="SCLWEB">This is a info 1</INFO>
     <WARN TIME="12/12/12 17:00:00.1" SOURCE="SCLWEB">This is a warning 1</WARN>
     <INFO TIME="12/12/12 13:00:00.1" SOURCE="SCLWEB">This is a info 2</INFO>
     <WARN TIME="12/12/12 17:00:00.1" SOURCE="SCLWEB">This is a warning 2</WARN>
     <INFO TIME="12/12/12 14:00:00.1" SOURCE="SCLWEB">This is a info 3</INFO>
     <WARN TIME="12/12/12 17:00:00.1" SOURCE="SCLWEB">This is a warning 3</WARN>
     <ALRT TIME="12/12/12 17:00:00.1" SOURCE="SCLWEB">This is a alert 1</ALRT>
     <INFO TIME="12/12/12 15:00:00.1" SOURCE="SCLWEB">This is a info 4</INFO>
     <WARN TIME="12/12/12 17:00:00.1" SOURCE="SCLWEB">This is a warning 4</WARN>
     <INFO TIME="12/12/12 16:00:00.1" SOURCE="SCLRTE">This is a info 5</INFO>
     <WARN TIME="12/12/12 17:00:00.1" SOURCE="SCLRTE">This is a warning 5</WARN>
     <INFO TIME="12/12/12 17:00:00.1" SOURCE="SCLRTE">This is a test 6</INFO>
     <SML TIME="12/12/12 18:00:00.2" SOURCE="SCLWEB">
        <CMDS NAME="swb">
            <OPCODE>12</OPCODE>
            <OBJID>23</OBJID>
        </CMDS>
     </SML>
     <SML TIME="12/12/12 18:00:00.2" SOURCE="SCLWEB">
        <SCRIPT NAME="scrub">
            <PARAM1>12</PARAM1>
            <PARAM2>23</PARAM2>
        </SCRIPT>
     </SML>
     <SML TIME="12/12/12 18:00:00.2" SOURCE="SCLWEB">
        <MESSAGE NAME="scl_do">
            <cmd>set relay1 to on</cmd>
        </MESSAGE>
     </SML>
</LOGFILE>
```

8.8 Appendix H. SML Schema

```
<?xml version="1.0" encoding="UTF-8" ?>

<SCHEMA NAME=”SML”
                xmlns="urn:schemas-microsoft-com:xml-data"
                xmlns:dt="urn:schemas-microsoft-com:datatypes" >

       <!-- definition of the method property used in computed fields →
       <OBJECT NAME=”METHOD”>
           <PROPERTY NAME=”PARMS” TYPE=”string” />
       </OBJECT>

       <!-- definition of the Discrete range property used for discrete fields →
       <OBJECT NAME=”OPTION”>
           <PROPERTY NAME=”VALUE” TYPE=”string” />
           <PROPERTY NAME=”STATE” TYPE=”string” />
       </OBJECT>

       <OBJECT NAME=”DRANGE”>
           <CONTAINS NAME=”OPTION” />
       </OBJECT>

       <!—- Definition of object field →
       <OBJECT NAME=”FIELD” >
           <PROPERTY NAME="MODE"
                VALUES="STATIC DYNAMIC" REQUIRED="NO" DEFAULT="DYNAMIC" />
           <PROPERTY NAME="VISIBLE"
```

```
          VALUES="YES NO" REQUIRED="NO" DEFAULT="YES" />
    <PROPERTY NAME="BIGENDIAN"
          VALUES="YES NO" REQUIRED="NO" DEFAULT="YES" />
    <PROPERTY NAME="UNITS" TYPE="string" REQUIRED="NO" />
    <PROPERTY NAME="NBITS" TYPE="int" REQUIRED="NO"  VISIBLE="YES"/>
    <PROPERTY NAME=”AS”TYPE=”string” />
    <PROPERTY NAME=”METHOD” />
</OBJECT>

<FIELD NAME=”INTEGER” >
    <PROPERTY NAME="MIN" TYPE="int" REQUIRED="NO" DEFAULT="0" />
    <PROPERTY NAME="MAX" TYPE="int" REQUIRED="NO" DEFAULT="0" />
    <PROPERTY NAME="MASK" TYPE="hex" REQUIRED="NO" DEFAULT="0" />
    <PROPERTY NAME=”DRANGE” />
</FIELD>

<INTEGER NAME="UB"
    TYPE="ui1"
    DESC=”Unsigned Byte”
    NBITS=”8”
    MIN=”0”
    MAX=”255” />

<INTEGER NAME="SB”
    TYPE=”ui1”
    DESC=”Signed Byte”
    NBITS=”8”
    MIN=”-127”
    MAX=”127” />

<INTEGER NAME="UI”
    TYPE=”ui2”
    DESC=”Unsigned Integer”
    NBITS=”16”
    MIN=”0”
    MAX=”65535” />

<INTEGER NAME="SI”
    TYPE=”ui2”
    DESC=”Signed Integer”
    NBITS=”16”
    MIN=”-32767”
    MAX=”32767” />

<INTEGER NAME="ULI”
    TYPE=”ui4”
    DESC=”Unsigned Long Integer”
    NBITS=”32”
    MIN=”0”
    MAX=”65535” />

<INTEGER NAME="SLI”
    TYPE=”ui4”
    DESC=”Signed Long Integer”
    NBITS=”32”
    MIN=”-65535”
    MAX=”65535” />

<!-- REAL fields -->

<FIELD NAME=”REAL” >
    <PROPERTY NAME="MIN"  TYPE="r8" REQUIRED="NO" DEFAULT="0" />
    <PROPERTY NAME="MAX" TYPE="r8" REQUIRED="NO" DEFAULT="0" />
    <PROPERTY NAME="RHIGH" TYPE="r8" REQUIRED="NO" DEFAULT="0" />
    <PROPERTY NAME="YHIGH" TYPE="r8" REQUIRED="NO" DEFAULT="0" />
    <PROPERTY NAME="YLOW" TYPE="r8" REQUIRED="NO" DEFAULT="0" />
    <PROPERTY NAME="RLOW" TYPE="r8" REQUIRED="NO" DEFAULT="0" />
</FIELD>

<!-- Single Floating Point ======================================== -->
<REAL NAME="SFP"
```

```
    TYPE="r4"
    DESC=”Single Floating Point”
    NBITS=”32”
    MIN=”dt:min”
    MAX=”dt:max”

    RHIGH=”dt:max”
    YHIGH=”dt:max”
    YLOW=”dt:min”
    RLOW=”dt:min” />

<!-- Double Floating Point ======================================== -->
<REAL NAME="DFP"
    TYPE="r8"
    DESC=”Double Floating Point”
    NBITS=”64”
    MIN=”dt:min”
    MAX=”dt:max”

    RHIGH=”dt:max”
    YHIGH=”dt:max”
    YLOW=”dt:min”
    RLOW=”dt:min” />

<!-- String ======================================================= -->
<FIELD NAME="STRING" TYPE="string" />


<!-- TIME (for now)================================================ -->
<FIELD NAME="TIME" TYPE="string" />


<!-- Aggregate Fields ============================================== →
<FIELD NAME=”BITFIELD” >
    <CONTAINS NAME=”FIELD” />
</FIELD>

<FIELD NAME=”UNION” >
    <CONTAINS NAME=”FIELD” />
</FIELD>

<FIELD NAME=”RECORD” >
    <CONTAINS NAME=”FIELD” />
</FIELD>

<FIELD NAME=”ARRAY” >
    <PROPERTY NAME=”DIM” TYPE=”int” />

    <CONTAINS NAME=”FIELD” />
</FIELD>


<!-- CMDS definition ============================================== -->
<OBJECT NAME="CMDS" >
    <PROPERTY NAME="TYPE" REQUIRED="NO"
         VALUES="NORMAL CRITICAL HAZARDOUS" DEFAULT="NORMAL" />
    <PROPERTY NAME="ABSTRACT" VALUES="YES NO" DEFAULT="NO" REQUIRED="NO" />

    <CONTAINS NAME=”FIELD” />
</OBJECT >

<!-- PACKETS definition =========================================== -->
<OBJECT NAME="PACKET" >
    <CONTAINS NAME="FIELD" />
</OBJECT >

<!-- MESSAGES definition ========================================== -->
<OBJECT NAME="NOTIFY" >
    <CONTAINS NAME="FIELD" />
</OBJECT >
```

```
        <OBJECT NAME="REQUEST" >
            <CONTAINS NAME="FIELD" />
        </OBJECT >

        <OBJECT NAME="REPLY" >
            <CONTAINS NAME="FIELD" />
        </OBJECT >

        <OBJECT NAME=”MESSAGE” />

        <MESSAGE NAME="NOTIFY_MESSAGE" >
            <CONTAINS NAME="NOTIFY" />
        </MESSAGE >

        <MESSAGE NAME="REQUEST_MESSAGE" >
            <CONTAINS NAME="REQUEST" />
            <CONTAINS NAME="REPLY" />
        </MESSAGE >

        <!-- Scripting Support ============================================= -->
        <OBJECT NAME="SCRIPT" >

            <PROPERTY NAME="LANGUAGE" TYPE="string" REQUIRED="NO" />
            <PROPERTY NAME="SOURCE" TYPE="string" REQUIRED="NO" />

            <CONTAINS ELEMENT="FIELD" />
        </OBJECT >

        <!-- SECTION definition ============================================= -->
        <OBJECT NAME="SECTION" >
            <CONTAINS NAME="CMDS" />
            <CONTAINS NAME="PACKET" />
            <CONTAINS NAME="MESSAGE" />
            <CONTAINS NAME="SCRIPT" />
            <CONTAINS NAME="RECORD" />
        </OBJECT >

        <!-- DATABASE definition ============================================= -->
        <OBJECT NAME="DATABASE"  >
            <PROPERTY NAME="VERSION" TYPE="string" REQUIRED="NO" />

            <CONTAINS NAME="SECTION" />
            <CONTAINS NAME="SCHEMA" />
        </OBJECT >

</SCHEMA>
```

8.9 Appendix I. References

Extract from *Abstract Syntax Notation One (ASN.1) - The Tutorial and Reference* by Doug Steedman **http://www.techapps.co.uk/asn1gloss.html**

The ASIST Spacecraft Ground System:
http://rs733.gsfc.nasa.gov/ASIST/ASIST-home.html

World wide web consortium

http://www.w3.org
http://www.w3.org/XML/
www.xml.org
www.xml.com
www.xdev.datachannel.com
http://msdn.microsoft.com/xml

BizTalk™ Framework Document Design Guide
http://www.biztalk.org/Resources/frame081.asp

BizTalk Framework XML Tags Specification
http://www.biztalk.org/Resources/tags081.asp

9 Index