The Consultative Committee for Space Data Systems

Report Concerning Space Data System Standards

XML TELEMETRIC AND COMMAND EXCHANGE (XTCE)

INFORMATIONAL REPORT

CCSDS 660.0-G-1

GREEN BOOK

July 2006

AUTHORITY

Issue:	Informational Report, Issue 1
Date:	July 2006
Location:	Not Applicable

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and reflects the consensus of technical panel experts from CCSDS Member Agencies. The procedure for review and authorization of CCSDS Reports is detailed in the *Procedures Manual for the Consultative Committee for Space Data Systems*.

This document is published and maintained by:

CCSDS Secretariat
Office of Space Communication (Code M-3)
National Aeronautics and Space Administration
Washington, DC 20546, USA

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification to this Report may occur. This Report is therefore subject to CCSDS document management and change control procedures, which are defined in the *Procedures Manual for the Consultative Committee for Space Data Systems*. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this Report should be addressed to the CCSDS Secretariat at the address on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (Roskosmos)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Korea Aerospace Research Institute (KARI)/Korea.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Organization (NSPO)/Taipei.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status/Remarks
CCSDS 660.0-G-1	XML Telemetric and Command Exchange (XTCE), Informational Report, Issue 1	July 2006	Current issue

CONTENTS

CONTENTS (continued)

Figure Page

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

This CCSDS Green Book presents a set of concepts and tutorial on a telemetry and telecommand database format for spacecraft monitoring and control. It has been prepared by the Spacecraft Monitoring and Control working group of the CCSDS Mission Operations and Information Management Systems (MOIMS) area.

The purpose of this document is to provide a tutorial for the XML Telemetric and Command Exchange (XTCE) standard whose goal is to facilitate the exchange of spacecraft telemetry and command databases between different organizations and systems during any mission phase. Such a non-proprietary format avoids the need for customized import/export tools, and the validation and new implementation of mission databases, which are often error-prone.

The scope of this document is limited to the exchange of satellite telemetry and commanding databases during the development and operation phases of a mission; it addresses general XTCE concepts and terminology. Finally, it provides a high-level overview of the XTCE Schema.

1.2 DOCUMENT STRUCTURE

Section 1 specifies the purpose, scope, and content of this document and specifies references to other documents that contain relevant material.

Section 2 provides a general description of the XTCE. This includes the following XTCE topics: overview, philosophy, concepts, and services.

Section 3 specifies the high-level exchange format or schema organization and structure.

Section 4 explains the major elements with examples.

Annex A shows a complete XTCE example.

Annex B is the glossary that contains definition of terms and acronyms.

1.3 GENERAL REFERENCES

The following websites contain important and interesting resource related to XML:

- **OMG XTCE website**- http://www.omg.org/space/xtce;
- **OMG Issues Page** - http://www.omg.org/technology/agreement.htm;
- **OMG Space Domain Task Force** - http://www.omg.org/space;
- **XML Tutorial** - http://www.w3schools.com/xml/default.asp;

- **XML Schema Tutorial -** http://www.w3schools.com/schema/default.asp;
- **Apache Xerces** - http://xml.apache.org/#xerces;
- **Castor -** http://www.castor.org/.

2 CONTEXT AND OVERVIEW

This document presents a non-proprietary format to describe a telemetry and telecommand database. This format is used by any space element that has to convey information containing telecommanding or/and telemetry to another space element. The XTCE standard is an OMG Space Task Force product that has been adopted by the CCSDS MOIMS group.

The OMG implementation of XTCE covers several more topics than does the CCSDS XTCE standards. The CCSDS XTCE Magenta Book will cover only CCSDS-based mission usage of XTCE.

2.1 WHY USE XTCE

XTCE is a solution to exchange telemetry and telecommand databases during the development and operation phases of a mission. It can be used to exchange a database between spacecraft manufacturers, instrument manufacturers, and different systems of the ground segment as seen in figure 2-1. It can also be used as an exchange mechanism between different development teams or between missions, which enhances database reuse.

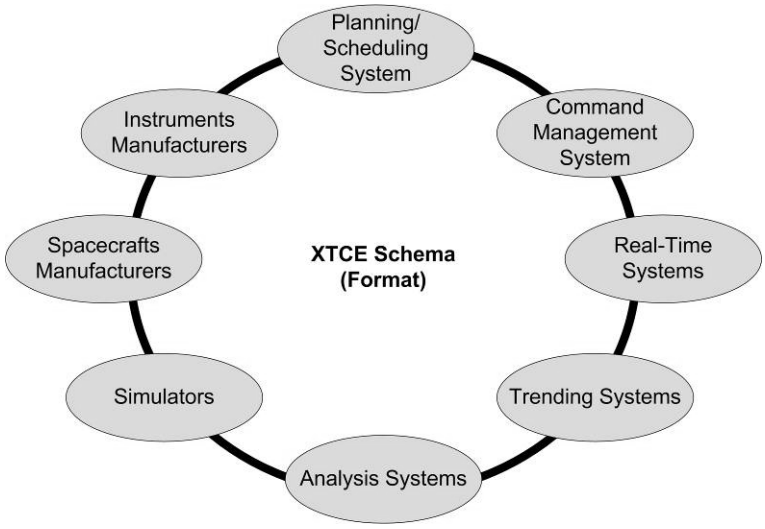

Figure 2-1: XTCE Exchange Concept

Eventually the usage of XTCE will drastically reduce mission costs for several reasons. First, there will be no need to create proprietary tools to convert and edit the database. This simple but error-prone task can then be avoided, as can the dependant validation and cross-checking that has to be done while converting data. Legacy telemetry and telecommand databases are easily converted to the XTCE format and those converters can make good use of the XML technologies support available in several programming languages. In addition, the use of a standard exchange format from the early stages of spacecraft development through mission operation will reduce life-cycle cost. Finally, XTCE will enable the user to

define one unique, non-proprietary format to describe a TM and TC database; this is useful when writing a mission database requirements document.

2.2 XTCE ORGANIZATION PHILOSOPHY

The organization of the XTCE schema reflects the pattern of a command and telemetry database. This database information is represented in its entirety in XML. XML is an easy and flexible way to add meaning and semantic to data. An XML file can constrain the contents of another XML file, or of itself. Such files are described using the XML Schema language, and XTCE is written in that language. This idea of an XML Schema is the backbone of XTCE.

XML Schema allows a hierarchical description of data. A telemetry and telecommand database written in XTCE will then be an XML file valid against the XTCE Schema. For recall, XML is a markup language where tags are defined by the user, in our case tags are defined in the schema. A variety of open-source or commercial programs are available to validate XML files, so the usage of XTCE will not necessarily lead to the added usage of commercial tools.

Section 3 gives a high level description of the XTCE Schema and its main structure without diving into too many details (For the details, please refer to the XTCE Magenta Book).

2.3 XTCE HIERARCHY CONCEPT

The main concept introduced in XTCE is the concept of a hierarchy of information. The information can be any part of a spacecraft (an instrument), the spacecraft itself, a constellation of spacecrafts or a ground station. This concept allows for the division of the information into smaller distinct components.

The most powerful usage of this concept, for satellite development, would be to create a separate telemetry and telecommand database in XTCE for each instrument on board the spacecraft, and another one for the spacecraft commanding itself (for navigation, attitude control, etc.). XTCE allows assembling these separate databases inputs into one large database (for instance, the satellite database) without integration problems. In this way, each dedicated database can describe its own information for telemetry or/and telecommanding. The Magenta Book (MB) provides examples and the Red Book (RB) provides definitions for the names and structure. The different databases will use the MB and RB standard to be consistent. For more details about the concepts, please refer to 3.1.

2.4 TELEMETRY

Telemetry containers allow users to describe how their telemetry is packaged. XTCE uses containers to describe a sequence of bits. In CCSDS, this packaging takes the form of a packet or even a part of a packet. For TDM, this packaging is a frame.

Parameters can be grouped together to build containers. Once containers are described the XTCE user can begin to define the properties.

Properties are encoding, decoding, and frame synchronization. Aside from these, XTCE allows a dedicated element to gather filter on containers. This means that XTCE support the concept of services and messages, and help in defining criterion for packet selection.

The user can also make reference to other algorithms not defined in XTCE for special purposes.

2.5 COMMANDING

Command containers allow users to describe how their commands are packaged and the actual order of arguments and parameters. The command and telemetry containers are identical except for some additional elements in the command container.

Some of the additional command definition elements allowed are:

- transmission constraint checks;
- verifiers, i.e., check based on telemetry after the supposed execution of a command;
- interlocks;
- significance (criticality of the command).

In XTCE, command arguments are local to a command and cannot be reused between commands. However, commands can be inherited, namely a command can be derived from another command. XTCE also supports a simple structure to describe a sequence of commands (a list of commands).

2.6 SERVICES

Services represent an advanced concept of selection in XTCE. This concept allows the grouping of containers under a given label. For example, all packets that have a dedicated parameter value can be put under the ‘memory dump’ service. The selection can be done on several parameters values.

2.7 HOW TO HANDLE ITEMS NOT ADDRESSED BY XTCE

XTCE is a highly generic schema where a lot of information can fit in, but it may happen that an exotic TM & TC architecture needs more. There are basically three ways to handle items not addressed by XTCE:

- Extend XTCE with a new schema.
- Overload something in XTCE for some other purpose that is not being used. There exist some fields in XTCE that can be used for different purposes. This is basically a practice that should be avoided but can still do the job.
- Create a separate XML Schema to describe additional data.

3 XTCE SCHEMA ORGANIZATION AND STRUCTURE

The XTCE Schema describes a hierarchy of databases for use at all levels from the overall spacecraft down to each individual instrument, for instance. A hierarchical system allows for an easier exchange of databases among the different levels. Each element in XTCE is known as a space system. These space systems describe the related telemetry and telecommand metadata as well as a dedicated header and services (figure 3-1), which will be covered later. In the following figures, dark colours mean parent elements, whereas lighter colours mean children elements.

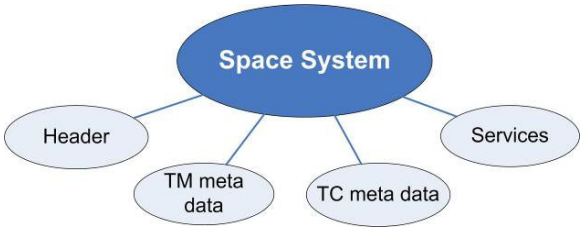

Figure 3-1: A Space System

All implementations of XTCE start out with a root element and can propagate through to every level of the overall space system. What makes the system robust is that, at each level, every element is a space system in and of itself (figure 3-2). This system enables the user to describe spacecraft telemetry and commanding step by step and regroup telemetry and telecommand data under logical systems.

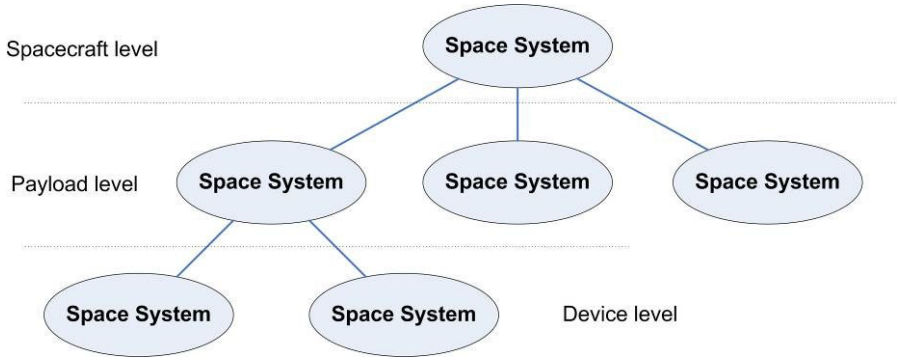

Figure 3-2: Hierarchy of Space Systems

The hierarchy of the space systems can contain as many levels as needed or be as simple as containing just one element (the root node). This flexibility applies to the individual space system as well. Each space system can contain as little or as many child elements (header, telemetry metadata and telecommand metadata, etc.) as needed. For example, one space system may only need to describe the telemetry metadata element and header for a space system. In this case the telecommand metadata would not be included.

3.1 SPACE SYSTEM—ROOT

Each collection of space systems starts off with a root space system and can contain as many children and grandchildren as needed. The hierarchical organization of the data is meant to be similar to the structure of a real space system and offers several important advantages over a flat representation:

- **Fewer name space collisions**—Almost every spacecraft contains redundant components for reliability. A communications spacecraft may have a dozen transponders each with the same set of telemetry points and commands. In a flat namespace, each of those telemetry points needs to be mapped to a unique name. Using a hierarchical namespace, those identical telemetry points can be simply placed into separate sub-space systems.
- **Better organization**—Modern spacecraft typically have thousands of commands and tens of thousands of telemetry parameters. The directory structure presented in XTCE provides an improved way to manage this large volume of data. Each subsystem developer can deliver space systems representing their subsystem without integration issues.
- **Natural hierarchy**—The hierarchical organization of the directory structure allows for the hierarchy of a spacecraft comprised of systems of systems as well as a **constellation of spacecraft space systems**.

Apart from containing children elements, the space system can contain a name, descriptions and aliases for human readability purposes. It is important to note that most elements in XTCE can contain ‘documentation’ information, like descriptions (usually a short and a long description), a name and several aliases if needed.

3.2 HEADER

The space system header describes information related to the XTCE file itself. It contains information about:

- date, version, classification, validation status (XML is validated, tested or drafted);
- a list of authors, name and surname of contributors;
- a list of notes, any textual notes can be written in the header;
- a list of history records, that can be used for configuration management purposes.

3.3 TELEMETRY METADATA

Since telemetry and command databases are frequently developed and maintained independently, the XTCE format separates this data but keeps their structure similar. Hence, a person that knows how to define telemetry metadata (figure 3-3) will not get lost describing telecommand metadata.

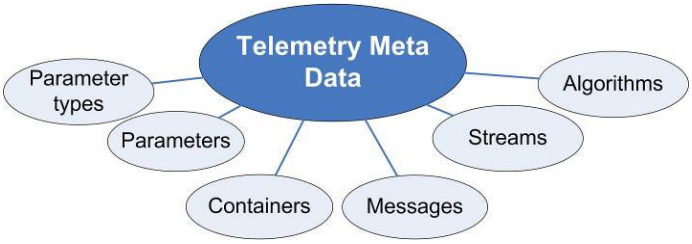

Figure 3-3: XTCE Telemetry

Telemetry metadata consists of different sets of data. First of all, common telemetry parameters must be defined in the parameter set, but a parameter must always make reference to a parameter type (such as integer, decimal, string, etc.).

Following the definition of a parameter, the next step is to define containers. Containers package data and are used to describe packets, TDM frame, frames that encapsulate packet, parts of packets or frames.

With parameters and containers, it is then possible to define different streams. A stream is usually a data link where frame synchronization is defined.

The two remaining elements, messages and algorithms, will be discussed in 3.3.4 and 3.3.6.

As for the space system, none of the child element of the telemetry metadata is mandatory. For example, if the user does not need messages, then messages can be omitted, and the XML file remains valid.

3.3.1 PARAMETER TYPE SET

A parameter type is the metadata for one or several parameters. A parameter will always need to be of a certain type. Typically a parameter type defines:

- data type;
- description of the type;
- alarms associated to the type;
- units;
- calibrations dedicated to this parameter type.

A calibration can be applied on numerical or textual data type such as a spline, textual or polynomial calibration.

Most parameters are telemetered, which means their values come from remote sensor measurements onboard a spacecraft. These parameters usually need to be encoded for proper

transmission through a channel. The related encoding data is part of the parameter type. Typical encoding data is the size in bits of the parameter, the byte order and parity checks.

3.3.2 PARAMETER SET

The parameter set is a collection of parameters. As explained in 3.3.1, a parameter always needs to reference a parameter type. Parameters usually consist of a name and a reference to a parameter type. A parameter can also have dedicated properties, not directly linked to its type definition.

3.3.3 CONTAINER SET

The container set is a collection of sequence containers. Sequence containers are extremely generic objects dedicated to package other objects. For example, containers are used to define TDM telemetry streams, packetised streams, or any other stream format.

In the case of a packetised stream, a container can be a frame, a packet, part of a frame or part of a packet. Conceptually, a container is a sequence of elements. Those elements are:

- raw parameters, or segment of parameters;
- stream segments;
- other containers or segment of containers.

The last bullet point indicates that containers can be inherited. The inheritance aspect of sequence containers is not only useful for minimizing the effort required to describe a family of containers, but also to express means of container identification—the process of distinguishing one container from others (e.g., distinguishing an abstract telemetry frame container from the telemetry frame header).

Sequence container inheritance may be arbitrarily deep. A container may represent a packet, a frame, a sub-frame or any other grouping/structure of data items. The simplest form of a sequence container is an ordered set of parameter references or other container references.

Figure 3-4 represents the concepts exposed above with the use of CCSDS Recommended Standards for Packet Telemetry (CCSDS 132.0-B-1 and CCSDS 133.0-B-1). In the context of XTCE, each group of rectangles is a sequence container. Light rectangles correspond to parameter entries, and the darker ones to sequence container entries for a dedicated container. One can see that the structure of frames, built on top of packets, is straightforward with the use of inheritance.

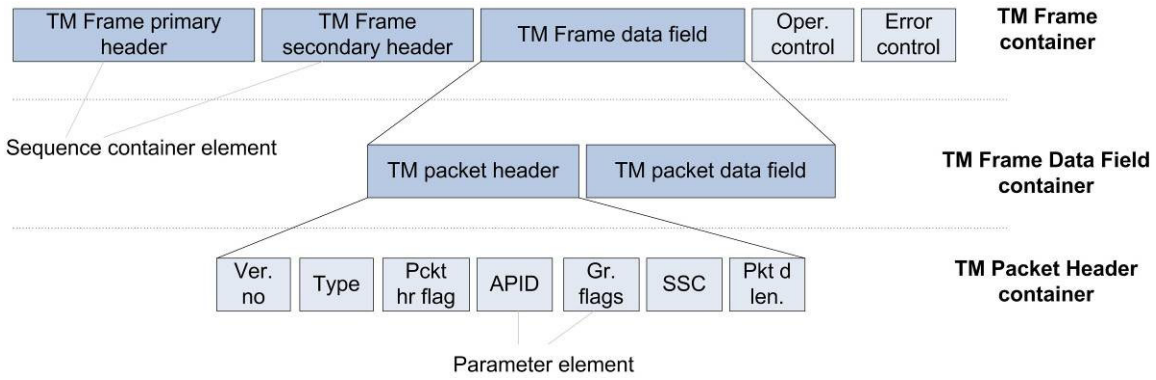

Figure 3-4: XTCE Container Inheritance Example

3.3.4 MESSAGE SET

The message set contains messages that are used to define services (discussed in 3.5) at the space system level. Messages can be defined and reused to aid in the definition of a service or thought of as a complementary alternative to service definitions. Defining messages under the message set element helps in readability and clarity.

A message is a container that is matched against a defined criterion. By this process it is possible to say that the message ‘housekeeping’ is attributed to all frames (in XTCE the container defined by the user) with the parameter ‘ABC’ set to ‘0’.

The concept of message is powerful as it allows to select containers (e.g., packets) based on a defined value. For example, the selection can be made on an application process identifier (APID) value, a header, a dedicated parameter or several parameter values in a container.

3.3.5 STREAM SET

A stream represents a communication link. In the context of a spacecraft, the stream will describe the data link from the spacecraft to the ground. There can be any number of stream definitions for telemetry including none at all.

A stream defines the processing needed to extract information out of the physical channel such as bit patterns for synchronization, expected rate, expected container (i.e., frame).

Streams can be of three types:

– fixed frame stream;

– variable frame stream;

– custom frame stream.

The first two streams are the most common, but if it is not enough, the user can define their own stream via the custom frame stream element.

A stream can be bound to a container or indirectly to a service (which itself is bound to some containers, see 3.5). Figure 3-5 presents the two possibilities to create a stream for Virtual Channel 0.

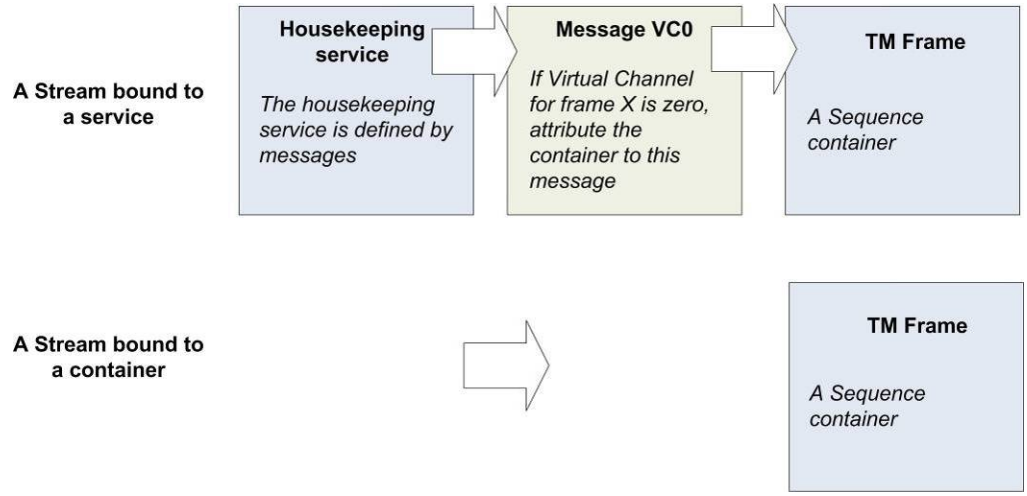

Figure 3-5: XTCE Stream Association

The stream definition contains the information to assemble and disassemble information transmitted through the channel.

3.3.6 ALGORITHM SET

The algorithm set contains a collection of algorithms that may be needed by space ground systems to perform some specialized processing to process the telemetry. There are a number of predefined algorithms and the algorithm section makes it possible to reference externally defined algorithms for arbitrarily sophisticated data processing.

3.4 COMMAND METADATA

The command metadata element is an extension of the telemetry metadata element (3.3). Command metadata contains the same elements as telemetry metadata with the inclusion of dedicated elements for pure telecommanding usage. Command metadata does not define messages but still supports the concept of filtering through the service concept at space system level. For parameter types, parameters, streams and algorithms, please refer to the appropriate subsection in the Telemetry Metadata section.

The dedicated elements are the argument type set and the meta-command set. Argument types defined in the argument type set are used to define meta-commands. A command is built with arguments and then packaged in a container.

The meta-command elements will regroup all commands dedicated to the space system. Figure 3-6 is a representation of the elements attached to the command metadata element.

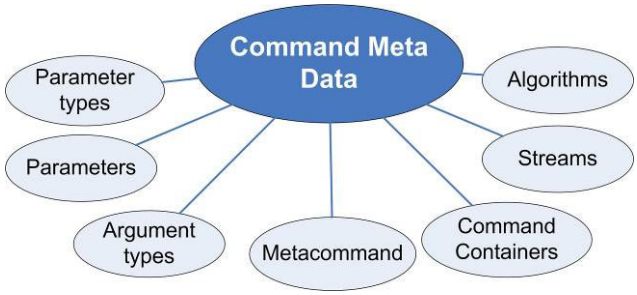

Figure 3-6: XTCE Command Metadata

3.4.1 ARGUMENT TYPE SET

The argument type set is very similar to the parameter type set, except that arguments that will be instantiated afterwards are strictly dedicated to one telecommand. Usually an argument expects to receive its value from a human actor (a spacecraft controller for example) or directly from a control system.

An argument type contains description of something that can have a value and is used as an operator supplied option to a command (i.e., a command argument). Information contained in an argument type includes:

- data type;
- textual description;
- a valid range;
- engineering unit;
- string conversion specifications and calibrations.

Most arguments are sent via a data link and must also include information about how the value is encoded for transmission. This information is also contained in the argument type and includes the size in bits, byte order, and parity checks.

3.4.2 META COMMAND SET

The meta-command set is where meta-commands are built and stored in the XTCE file. A meta-command is built out of an argument list, a container, some transmission constraints, significance, interlock, verifiers and parameters (figure 3-7).

Simple command sequences can also be packaged in the meta-command set as a Block meta-command element. Moreover, meta-commands defined externally can be reused in this set by the use of references.

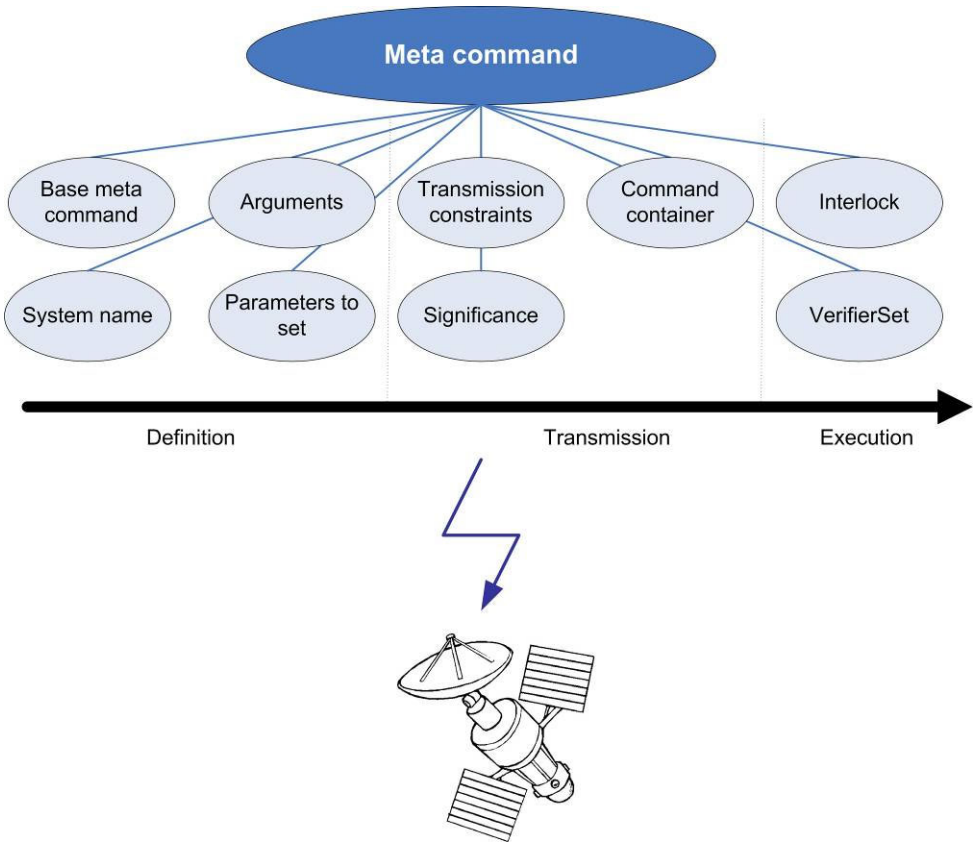

Figure 3-7: Overview of a Meta-Command

3.4.2.1 Base Meta Command

A meta-command can be derived from another meta-command, known as base meta-commands. Arguments of the base meta-command are then re-specified for the current meta-command.

3.4.2.2 Argument List

The argument list of the meta-command describes arguments used by the command. Command arguments can be of any standard data types and should be instantiated by an argument type (defined in the command metadata element).

3.4.2.3 Command Container

The command container describes how the command is packaged. The concept here is similar to the one used with telemetry packaging. Except that here command containers may also have arguments entries that are not allowed in a sequence container (i.e., 'telemetry container'). Moreover command container can also support fixed value entries.

The inheritance mechanism also applies to command containers. Therefore, it is intended to describe a generic container at the command metadata level (out of the meta-command element) and to reference it, either as a base container or as a container entry, to build the current container for the command.

The command container described under the meta-command element allows fixed values and arguments entries. It is important to note that these entries are not allowed for container defined under the command metadata element (at a higher level), as fixed values and arguments can only be referred in the context of a dedicated meta-command.

3.4.2.4 Transmission Constraint List

Transmission constraints are defined in a dedicated list. Usually constraints consist of comparing telemetry parameter values. All conditions defined in a transmission list must be met prior to the actual transmission of the command.

The constraints can also be represented as a custom algorithm, invoking any proprietary processing before the command transmission (so constraints are not only limited to parameters comparisons).

3.4.2.5 Default Significance and Context Significance List

Some command and control systems may require special user access or confirmations before transmitting commands. The significance includes the name of the space system at risk, and a significance level. Significance levels are: *none, watch, warning, distress, critical and severe*.

3.4.2.6 Interlock

An interlock is a constraint for commands following the current command being executed. It will block successive commands until the current command has reached a certain stage (through verification). Interlocks are scoped to a space system basis. The verification to wait for (to release the lock) is linked to the status of the command. Examples of stages are: *transferred to range, received, accepted, queued, executing, and complete*.

3.4.2.7 Verifier Set

A verifier is a conditional check on the telemetry down linked from the space system that provides indication on the execution of a command. There are eight different verifiers each associated with difference stages in command completion*: Transferred to range, Transferred from range, Received, Accepted, Queued, Execution, Complete, and Failed*. Verifiers are groups in a set for the current telecommand being defined.

3.4.2.8 Parameter to Set List

The parameter to set list contains parameters whose value is expected to be set after the current command has reached a given stage.

3.5 SERVICE SET

The service set is a collection of services. Services are used to define streams, but they can also help in containers selection based on one or several criterions (like parameter values checks).

Usually a service is composed of a set of messages, where each message selects containers according to one or several criterions. Services can also be defined without messages, but in that case the service has to make direct reference to one or several generic containers, with appropriate conditions in order to attribute them to a service. As messages are not present on the commanding side, referencing generic containers is the only way to define services for commanding.

The concept of services has been introduced in XTCE for the European space organizations that deal with services. The ECSS-E-70-41A (Packet utilization standard), compliant with CCSDS standards, goes a little bit deeper in packet composition and introduces the notion of services. Please refer to that standard in order to get examples and detailed definition of services.

A typical implementation of a service in XTCE will use messages to characterize containers (i.e., frames), and attribute them to a service at the space system level.

3.6 BIG PICTURE OF XTCE MAJOR ELEMENTS

Figure 3-8 presents a big picture to illustrate the hierarchy of the XTCE schema. All major elements are represented up to a depth of 3 (one colour per depth level). This is sufficient to understand the structure of XTCE Schema without diving into too many details.

In this figure, one can see that the telemetry metadata and the command metadata elements are similar, even though the command element contains more information than the telemetry metadata element.

Moreover, the definition of types for parameters and arguments are also similar at 95%. The same data types are allowed for each, and these types are very common but can also be customized by the XTCE user. Namely, the major data types in XTCE are:

- string, enumerated;
- integer, float, binary;
- Boolean;
- arrays;
- absolute and relative times.

3.7 XTCE XML TREE OVERVIEW

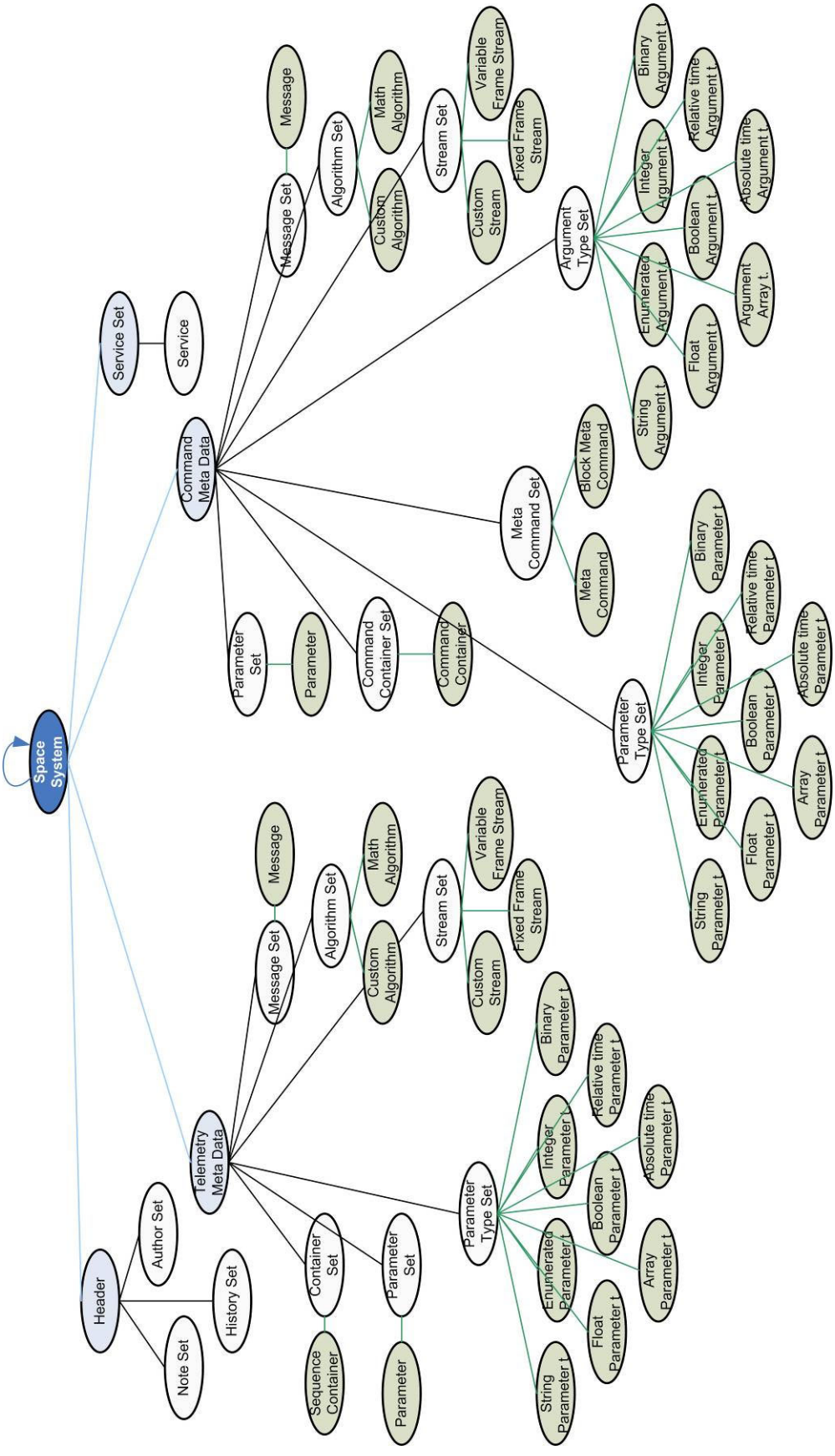

Figure 3-8: XTCE Tree Overview

4 TYPICAL USAGE SCENARIO OF XTCE

This section addresses the issue of a Spacecraft Project manager who gets various on-board subsystems and payloads for his spacecraft that are developed by various contractors. From each of them, he gets an XML file containing telecommand and telemetry for that subsystem/payload, which he then merges into the Spacecraft Source Database (DB) and uses for Assembly, Integration Verification (AIV). The Spacecraft Source DB is then passed to operations (Operational Database) where it is used by several ground applications.

4.1 XTCE USAGE SCENARIO

During a typical mission lifecycle, there are several levels of database integration, translation and re-translation. The spacecraft manufacturer integrates many systems, sub-systems, and sub-sub-systems. These sub-systems are developed by vendors or subcontractors from around the world. Each of these sub-systems will generate telemetry data and receive commands that must be described in a database. The spacecraft manufacturer must include all sub-system data in a master spacecraft database (Source Database). This master spacecraft Source Database will then be sent to the mission readiness team for inclusion or translation into many different databases used for operations (Operational Database). Figure 4-1 depicts a representative mission database flow from AIV to Operations. Today there is no standard telemetry and command database exchange format, and each of these database exchanges requires custom translations which incurs significant cost, schedule and risk to the mission. Using XTCE as the program's standard or common telemetry and command database exchange format will reduce mission cost, schedule, and risk by eliminating custom database translations. XTCE is the exchange of database information with other databases, not the direct interface for sending commands or receiving telemetry between systems and subsystems.

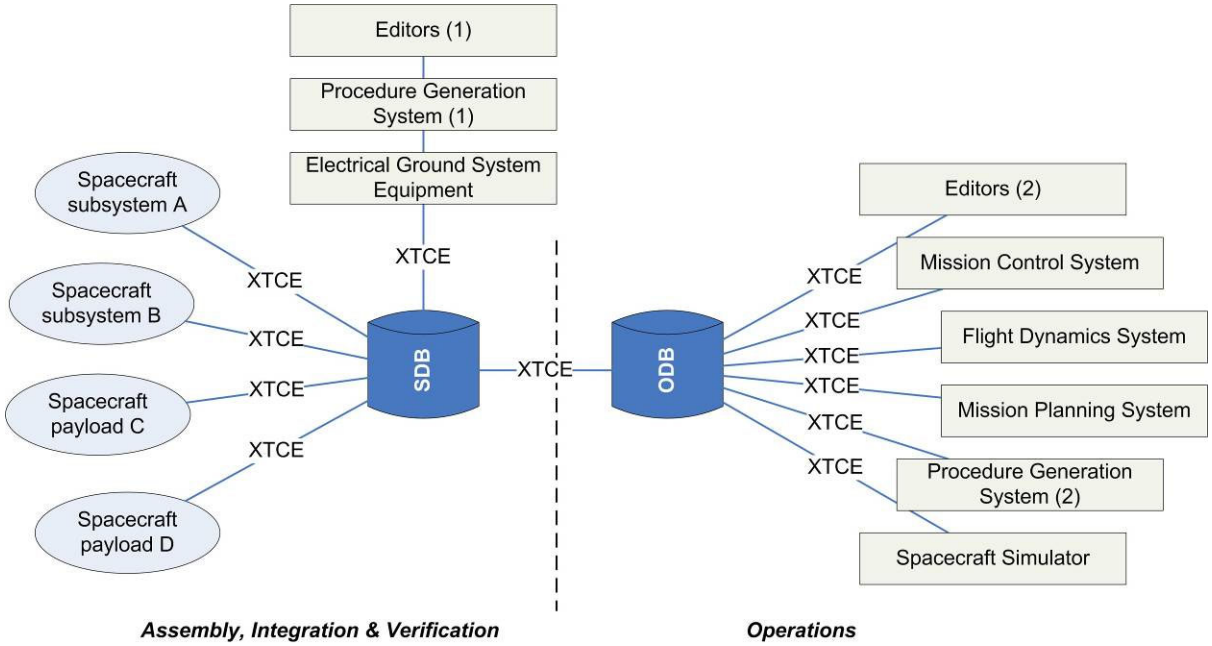

Figure 4-1: XTCE Usage Scenario

ANNEX A

XTCE EXAMPLE

The following is recommended steps to write an XTCE compliant XML file:

1. Create an XML file with the appropriate Space System element, and name the Space System element.

2. Define the Telemetry Parameter Types in the appropriate set.

3. Instantiate the Telemetry Parameters on top of Parameter Types. All the parameters are located in their own set (Parameter Set element).

4. Build Sequence Containers (packages for telemetry data).

5. Define Stream for Telemetry.

6. Define Telecommand Argument types (like point 2).

7. Define Meta Commands and their various checks and constraints.

8. Define Command Containers.

9. Define Stream for Telecommand.

A1 AN EXAMPLE OF A CCSDS XTCE XML FILE

Here is an example of two commands described in XTCE along with telemetry parameters. The following example has a large amount of lines to define CCSDS packaging (frames and packets), which are generic enough to be reused to specify telecommand packets and telemetry packets through inheritance.

A1.1 SPACE SYSTEM

The SpaceSystem is the root element of an XTCE XML file and has may attributes defining locations of other files. In particular, the space system makes reference to the XTCE schema (xsi:schemaLocation) and to the language used in the schema (xmlns:xsi), and of course to a dedicated namespace (xmlns).

In the following code fragment, the top level hierarchy appears with two distinct elements for telemetry metadata and command metadata, each containing their own empty elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<xtce:SpaceSystem xmlns:xtce="http://www.omg.org/space/xtce"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.omg.org/space/xtce:\DOCUME~1\jmuller\MYDOCU~1\De
velopment\Converters\XSD\SpaceSystemV1.1.xsd" name="XTCE GB example"
shortDescription="XTCE sample for CCSDS Green Book">

    <xtce:LongDescription>This file is intented to give a quick overview of XTCE for CCSDS
based mission. This examples does not cover all possibilities of XTCE but tries to give relevant
examples of the main points covered by the standard. </xtce:LongDescription>

    <xtce:Header validationStatus="Validated" date="2006-01-23" version="0.1"/>
    <TelemetryMetaData>
        <ParameterTypeSet>[...]</ParameterTypeSet>
        <ParameterSet>[...]</ParameterSet>
        <ContainerSet>[...]</ContainerSet>
        <StreamSet>[...]</StreamSet>
    </TelemetryMetaData>
    <CommandMetaData>
        <ArgumentTypeSet>[...]</ArgumentTypeSet>
        <MetaCommandSet>[...]</MetaCommandSet>
        <CommandContainerSet>[...]</CommandContainerSet>
        <StreamSet>[...]</StreamSet>
    </CommandMetaData>
</SpaceSystem>
```

A1.2 TELEMETRY METADATA

This step defines the three empty elements of the telemetry metadata element; parameter type set, parameter set and container set.

A1.2.1 Telemetry Parameter Type Set

First step is to define parameter types that will be used to instantiate parameters afterwards. Some generic integer types are used to declared parameters that will be used to define CCSDS frames and packets as XTCE has no previous knowledge of the packaging used.

```
<xtce:ParameterTypeSet>
    <!-- Parameter types used by frames and packets (no data parameter) -->
    <xtce:IntegerParameterType name="4ByteInteger" sizeInBits="32" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="2ByteInteger" sizeInBits="16" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="1ByteInteger" sizeInBits="8" signed="false">
        <xtce:UnitSet/>
```

```
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="14BitInteger" sizeInBits="14" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="11BitInteger" sizeInBits="11" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="10BitInteger" sizeInBits="10" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="6BitInteger" sizeInBits="6" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="4BitInteger" sizeInBits="4" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="3BitInteger" sizeInBits="3" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="2BitInteger" sizeInBits="2" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <xtce:IntegerParameterType name="1BitInteger" sizeInBits="1" signed="false">
        <xtce:UnitSet/>
    </xtce:IntegerParameterType>
    <!-- "data" parameter types references by telecommands-->
    <xtce:IntegerParameterType name="AST10061" signed="false" sizeInBits="1">
        <xtce:UnitSet/>
        <xtce:ToString>
            <xtce:RangeEnumeration label="ENABLED" maxInclusive="1" minInclusive="1"/>
            <xtce:RangeEnumeration label="DISABLED" maxInclusive="0" minInclusive="0"/>
        </xtce:ToString>
    </xtce:IntegerParameterType>
</xtce:ParameterTypeSet>
```

A1.2.2 Telemetry Parameter Set

The parameters are defined on top of parameter types in their own parameter set. They are given a name and a parameter type.

```
<xtce:ParameterSet>
    <!-- Parameter used by frames and packets (no data parameter) -->
    <xtce:Parameter parameterTypeRef="4ByteInteger" name="CCSDS_TF_OC"/>
    <xtce:Parameter parameterTypeRef="2ByteInteger" name="CCSDS_TF_EC"/>
    <xtce:Parameter parameterTypeRef="2BitInteger" name="CCSDS_FVERSION"/>
    <xtce:Parameter parameterTypeRef="10BitInteger" name="CCSDS_SC_ID"/>
    <xtce:Parameter parameterTypeRef="3BitInteger" name="CCSDS_TF_VC"/>
    <xtce:Parameter parameterTypeRef="1BitInteger" name="CCSDS_OP_CTRL"/>
    <xtce:Parameter parameterTypeRef="1ByteInteger" name="CCSDS_MS_CNT"/>
    <xtce:Parameter parameterTypeRef="1ByteInteger" name="CCSDS_VC_CNT"/>
```

```
    <xtce:Parameter parameterTypeRef="1BitInteger" name="CCSDS_TF_SECH"/>
    <xtce:Parameter parameterTypeRef="1BitInteger" name="CCSDS_TF_SYNC"/>
    <xtce:Parameter parameterTypeRef="1BitInteger" name="CCSDS_TF_ORDER"/>
    <xtce:Parameter parameterTypeRef="2BitInteger" name="CCSDS_TF_SEGM"/>
    <xtce:Parameter parameterTypeRef="1BitInteger" name="CCSDS_TF_FH"/>
    <xtce:Parameter parameterTypeRef="2BitInteger" name="CCSDS_TF_HV"/>
    <xtce:Parameter parameterTypeRef="6BitInteger" name="CCSDS_TF_HL"/>
    <xtce:Parameter parameterTypeRef="3BitInteger" name="CCSDS_VERSION"/>
    <xtce:Parameter parameterTypeRef="1BitInteger" name="CCSDS_TYPE"/>
    <xtce:Parameter parameterTypeRef="1BitInteger" name="CCSDS_SEC_HD"/>
    <xtce:Parameter parameterTypeRef="11BitInteger" name="CCSDS_APID"/>
    <xtce:Parameter parameterTypeRef="2BitInteger" name="CCSDS_GP_FLAGS"/>
    <xtce:Parameter parameterTypeRef="14BitInteger" name="CCSDS_SSC"/>
    <xtce:Parameter parameterTypeRef="2ByteInteger" name="CCSDS_PLENGTH"/>
    <xtce:Parameter parameterTypeRef="1BitInteger" name="CCSDS_BYPASS"/>
    <xtce:Parameter parameterTypeRef="1BitInteger" name="CCSDS_CCF"/>
    <xtce:Parameter parameterTypeRef="2BitInteger" name="CCSDS_SPARE"/>
    <xtce:Parameter parameterTypeRef="10BitInteger" name="CCSDS_SCT_ID"/>
    <xtce:Parameter parameterTypeRef="6BitInteger" name="CCSDS_VC_ID"/>
    <xtce:Parameter parameterTypeRef="10BitInteger" name="CCSDS_LENGTH"/>
    <xtce:Parameter parameterTypeRef="1ByteInteger" name="CCSDS_FSN"/>
    <xtce:Parameter parameterTypeRef="2ByteInteger" name="CCSDS_F_ERROR_CTRL"/>
    <xtce:Parameter parameterTypeRef="2BitInteger" name="CCSDS_SEQ_FLAG"/>
    <xtce:Parameter parameterTypeRef="6BitInteger" name="CCSDS_MAP_ID"/>

    <!-- "data" parameters -->
    <xtce:Parameter name="AST10061" parameterTypeRef="AST10061"
shortDescription="GND_UPDATE_ENA">
        <xtce:ParameterProperties dataSource="derived">
            <xtce:SystemName>112</xtce:SystemName>
        </xtce:ParameterProperties>
    </xtce:Parameter>
</xtce:ParameterSet>
```

A1.2.3 Telemetry Container Set

The telemetry container set defines the telemetry packaging, as this example is CCSDS based, frames and packets are described in details, with a parameter each. Containers are put together to build packets and frames.

```
<xtce:ContainerSet>
    <xtce:SequenceContainer name="TMFrame" shortDescription="CCSDS TM Frame">
        <xtce:EntryList>
            <xtce:ContainerRefEntry containerRef="AbstractTMFrameHeader"/>
            <xtce:ContainerRefEntry containerRef="AbstractTM Packet"/>
            <xtce:ContainerRefEntry containerRef="AbstractTMFrameTail "/>
        </xtce:EntryList>
    </xtce:SequenceContainer>
    <xtce:SequenceContainer name="AbstractTMFrameTail" shortDescription="CCSDS TM Frame
Tail" abstract="true">
```

```
        <xtce:EntryList>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_OC"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_EC"/>
        </xtce:EntryList>
    </xtce:SequenceContainer>
    <xtce:SequenceContainer name="AbstractTMFrameHeader" shortDescription="CCSDS TM
Frame Header" abstract="true">
        <xtce:EntryList>
            <xtce:ParameterRefEntry parameterRef="CCSDS_FVERSION"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_SC_ID"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_VC"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_OP_CTRL"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_MS_CNT"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_VC_CNT"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_SECH"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_SYNC"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_ORDER"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_SEGM"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_FH"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_HV"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TF_HL"/>
        </xtce:EntryList>
    </xtce:SequenceContainer>
    <xtce:SequenceContainer name="AbstractTM Packet " shortDescription="CCSDS TM Packet
Header" abstract="true">
        <xtce:EntryList>
            <xtce:ParameterRefEntry parameterRef="CCSDS_VERSION"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TYPE"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_SEC_HD"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_APID"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_GP_FLAGS"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_SSC"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_PLENGTH"/>
        </xtce:EntryList>
    </xtce:SequenceContainer>
    <xtce:SequenceContainer name="ComboPacket" shortDescription="contains all telemetred data
parameter for example purposes">
        <xtce:EntryList>
            <xtce:ParameterRefEntry parameterRef="AST10061"/>
        </xtce:EntryList>
        <xtce:BaseContainer containerRef="CCSDS TM Packet Header">
            <xtce:RestrictionCriteria>
                <xtce:ComparisonList>
                    <xtce:Comparison parameterRef="CCSDS_VERSION" value="0"/>
                    <xtce:Comparison parameterRef="CCSDS_TYPE" value="0"/>
                    <xtce:Comparison parameterRef="CCSDS_SEC_HD" value="0"/>
                    <xtce:Comparison parameterRef="CCSDS_APID" value="123"/>
                </xtce:ComparisonList>
            </xtce:RestrictionCriteria>
        </xtce:BaseContainer>
    </xtce:SequenceContainer>
</xtce:ContainerSet>
```

A1.2.4 Telemetry Stream

The telemetry streams is defined with a generic container, ie the telemetry frame defined before.

```
<xtce:StreamSet>
    <xtce:FixedFrameStream name="non turbo coded stream" shortDescription="This is an example
for CCSDS" frameLengthInBits="1784">
        <xtce:ContainerRef containerRef="TMFrame"/>
        <xtce:SyncStrategy>
            <xtce:SyncPattern patternLengthInBits="32" pattern="1ACFFC1D"/>
        </xtce:SyncStrategy>
    </xtce:FixedFrameStream>
</xtce:StreamSet>
```

A1.3 COMMAND METADATA

The command metadata element is composed of arguments, meta-commands and command containers.

A1.3.1 Command Argument Types

Arguments definition is similar to parameter type definitions, the same data types are available. In this case, only two arguments need to be defined. The argument types will be instantiated later when building meta-command. There is no Argument Set under the command metadata element, but there is an argument list for each meta-command.

```
<xtce:ArgumentTypeSet>
    <!-- Command ASC5054 Arguments -->
    <xtce:IntegerArgumentType name="ASP50092" shortDescription="Vel_Z" signed="true" sizeInBits="32">
        <xtce:UnitSet>
            <xtce:Unit>mm/s</xtce:Unit>
        </xtce:UnitSet>
    </xtce:IntegerArgumentType>
    <xtce:IntegerArgumentType name="ASP50091" shortDescription="Vel_Y" signed="true" sizeInBits="32">
        <xtce:UnitSet>
            <xtce:Unit>mm/s</xtce:Unit>
        </xtce:UnitSet>
    </xtce:IntegerArgumentType>
    <xtce:IntegerArgumentType name="ASP50090" shortDescription="Vel_X" signed="true" sizeInBits="32">
        <xtce:UnitSet>
            <xtce:Unit>mm/s</xtce:Unit>
        </xtce:UnitSet>
    </xtce:IntegerArgumentType>
    <xtce:IntegerArgumentType name="ASP50037" shortDescription="Pos_Z" signed="true" sizeInBits="32">
        <xtce:UnitSet>
            <xtce:Unit>cm</xtce:Unit>
        </xtce:UnitSet>
    </xtce:IntegerArgumentType>
```

```
    <xtce:IntegerArgumentType name="ASP50036" shortDescription="Pos_Y" signed="true" sizeInBits="32">
        <xtce:UnitSet>
            <xtce:Unit>cm</xtce:Unit>
        </xtce:UnitSet>
    </xtce:IntegerArgumentType>
    <xtce:IntegerArgumentType name="ASP50035" shortDescription="Pos_X" signed="true" sizeInBits="32">
        <xtce:UnitSet>
            <xtce:Unit>cm</xtce:Unit>
        </xtce:UnitSet>
    </xtce:IntegerArgumentType>
    <xtce:IntegerArgumentType name="ASP50089" shortDescription="Time_MS" signed="false" sizeInBits="32">
        <xtce:UnitSet>
            <xtce:Unit>msec</xtce:Unit>
        </xtce:UnitSet>
    </xtce:IntegerArgumentType>
    <xtce:IntegerArgumentType name="ASP50088" shortDescription="Time_JD" signed="false" sizeInBits="16">
        <xtce:UnitSet>
            <xtce:Unit>Day</xtce:Unit>
        </xtce:UnitSet>
    </xtce:IntegerArgumentType>
    <xtce:IntegerArgumentType name="ASP50202" shortDescription="ActivityID (dec)" signed="false" sizeInBits="8">
        <xtce:UnitSet/>
    </xtce:IntegerArgumentType>
    <xtce:IntegerArgumentType name="ASP50104" shortDescription="FunctionID (Hex)" signed="false" sizeInBits="8">
        <xtce:UnitSet/>
    </xtce:IntegerArgumentType>
    <xtce:IntegerArgumentType name="YSP50014" shortDescription="Function_ID" signed="false" sizeInBits="8">
        <xtce:UnitSet/>
    </xtce:IntegerArgumentType>
</xtce:ArgumentTypeSet>
```

A1.3.2 MetaCommand

The following examples describe two commands with several arguments. Arguments are instanciated on top of an argument type (described before).

```
<xtce:MetaCommandSet>
<xtce:MetaCommand name="ASC50541" shortDescription="Orbit_Update">
        <xtce:LongDescription>PerformActivityOfFunc_PID_10_FuncID_44_ActID_0</xtce:LongDescription>
    <xtce:AncillaryDataSet>
        <xtce:AncillaryData name="SubSystem">1</xtce:AncillaryData>
        <xtce:AncillaryData name="CommandType"/>
    </xtce:AncillaryDataSet>
    <xtce:ArgumentList>
        <xtce:Argument argumentTypeRef="ASP50092" name="ASP50092"/>
        <xtce:Argument argumentTypeRef="ASP50091" name="ASP50091"/>
        <xtce:Argument argumentTypeRef="ASP50090" name="ASP50090"/>
        <xtce:Argument argumentTypeRef="ASP50037" name="ASP50037"/>
        <xtce:Argument argumentTypeRef="ASP50036" name="ASP50036"/>
        <xtce:Argument argumentTypeRef="ASP50035" name="ASP50035"/>
```

```
        <xtce:Argument argumentTypeRef="ASP50089" name="ASP50089"/>
        <xtce:Argument argumentTypeRef="ASP50088" name="ASP50088"/>
        <xtce:Argument argumentTypeRef="ASP50202" name="ASP50202"/>
        <xtce:Argument argumentTypeRef="ASP50104" name="ASP50104"/>
    </xtce:ArgumentList>
    <xtce:CommandContainer name="ASC50541" shortDescription="Command ASC50541Packet">
        <xtce:EntryList>
            <xtce:ContainerRefEntry containerRef="AbstractTCPacket">
                <xtce:IncludeCondition>
                    <xtce:ComparisonList>
                        <xtce:Comparison parameterRef="CCSDS_APID" value="172"/>
                        <xtce:Comparison parameterRef="CCSDS_TYPE" value="1"/>
                        <xtce:Comparison parameterRef="CCSDS_SEC_HD" value="0"/>
                    </xtce:ComparisonList>
                </xtce:IncludeCondition>
            </xtce:ContainerRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50092">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>224</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50091">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>192</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50090">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>160</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50037">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>128</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50036">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>96</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50035">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>64</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50089">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>32</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50088">
```

```
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>16</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50202">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>8</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
            <xtce:ArgumentRefEntry argumentRef="ASP50104">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>0</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
        </xtce:EntryList>
    </xtce:CommandContainer>
    <xtce:DefaultSignificance consequenceLevel="none"/>
    <xtce:VerifierSet/>
</xtce:MetaCommand>
<xtce:MetaCommand name="ASC50439" shortDescription="Gnd_Orbit_Update Disable">
    <xtce:LongDescription>Deactivate_a_Function_PID_10_FuncID_44</xtce:LongDescription>
    <xtce:SystemName>1</xtce:SystemName>
    <xtce:ArgumentList>
        <xtce:Argument argumentTypeRef="YSP50014" name="YSP50014"/>
    </xtce:ArgumentList>
    <xtce:CommandContainer name="ASC50439" shortDescription="Command ASC50439 Packet">
        <xtce:EntryList>
            <xtce:ContainerRefEntry containerRef="AbstractTCPacket">
                <xtce:IncludeCondition>
                    <xtce:ComparisonList>
                        <xtce:Comparison parameterRef="CCSDS_APID" value="172"/>
                        <xtce:Comparison parameterRef="CCSDS_TYPE" value="1"/>
                        <xtce:Comparison parameterRef="CCSDS_SEC_HD" value="0"/>
                    </xtce:ComparisonList>
                </xtce:IncludeCondition>
            </xtce:ContainerRefEntry>
            <xtce:ArgumentRefEntry argumentRef="YSP50014">
                <xtce:LocationInContainerInBits referenceLocation="containerStart">
                    <xtce:FixedValue>0</xtce:FixedValue>
                </xtce:LocationInContainerInBits>
            </xtce:ArgumentRefEntry>
        </xtce:EntryList>
    </xtce:CommandContainer>
    <xtce:DefaultSignificance consequenceLevel="critical"/>
    <xtce:VerifierSet>
        <xtce:ExecutionVerifier>
                <xtce:Comparison parameterRef="AST10061" useCalibratedValue="true" value="DISABLED"/>
            <xtce:CheckWindow timeToStartChecking="PT0H0M0S" timeToStopChecking="PT0H0M10S"/>
        </xtce:ExecutionVerifier>
    </xtce:VerifierSet>
</xtce:MetaCommand>
</xtce:MetaCommandSet>
```

A1.3.3 Command Containers

Frames and packets for telecommanding.

```
<xtce:CommandContainerSet>
    <xtce:CommandContainer name="TCFrame" shortDescription="CCSDS TC Frame">
        <xtce:EntryList>
            <xtce:ContainerRefEntry containerRef="AbstractTCFrameHeader"/>
            <xtce:ContainerRefEntry containerRef="AbstractTCSegment"/>
            <xtce:ContainerRefEntry containerRef="AbstractTCFrameTail"/>
        </xtce:EntryList>
    </xtce:CommandContainer>
    <xtce:CommandContainer name="AbstractTCFrameHeader" shortDescription="CCSDS TC Frame
Header" abstract="true">
        <xtce:EntryList>
            <xtce:ParameterRefEntry parameterRef="CCSDS_FVERSION"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_BYPASS"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_CCF"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_SPARE"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_SCT_ID"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_VC_ID"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_LENGTH"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_FSN"/>
        </xtce:EntryList>
    </xtce:CommandContainer>
    <xtce:CommandContainer name="AbstractTCFrameTail" shortDescription="CCSDS TC Frame Tail" abstract="true">
        <xtce:EntryList>
            <xtce:ParameterRefEntry parameterRef="CCSDS_F_ERROR_CTRL"/>
        </xtce:EntryList>
    </xtce:CommandContainer>
    <xtce:CommandContainer name="AbstractTCSegment" shortDescription="CCSDS TC Segment" abstract="true">
        <xtce:EntryList>
            <xtce:ParameterRefEntry parameterRef="CCSDS_SEQ_FLAG"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_MAP_ID"/>
            <xtce:ContainerRefEntry containerRef="AbstractTCPacket"/>
        </xtce:EntryList>
    </xtce:CommandContainer>
    <xtce:CommandContainer name="AbstractTCPacket" shortDescription="CCSDS TC Packet Header"
abstract="true">
        <xtce:EntryList>
            <xtce:ParameterRefEntry parameterRef="CCSDS_VERSION"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_TYPE"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_SEC_HD"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_APID"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_GP_FLAGS"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_SSC"/>
            <xtce:ParameterRefEntry parameterRef="CCSDS_PLENGTH"/>
        </xtce:EntryList>
    </xtce:CommandContainer>
</xtce:CommandContainerSet>
```

A1.3.4 Telecommand Stream

```
<xtce:StreamSet>
    <xtce:VariableFrameStream name="Telecommand Link">
        <xtce:ContainerRef containerRef="TCFrame"/>
        <xtce:SyncStrategy>
            <xtce:Flag flagSizeInBits="6" flagBitType="ones"/>
        </xtce:SyncStrategy>
    </xtce:VariableFrameStream>
</xtce:StreamSet>
```

All previous code fragments put together give a complete XML files, that is valid against XTCE Schema 1.1. For further details, an XTCE developer can look for implementation details in the XTCE Magenta Book.

A2 QUICK NOTES TO HELP UNDERSTAND XML & XML SCHEMA

A2.1 SHORT TUTORIAL ON XML

The first line of an XML document is the XML declaration: it is an optional line stating what version of XML is in use (normally version 1.0), and may also contain information about character encoding and external dependencies.

The remainder of the document consists of nested elements, some of which have attributes and content. An element typically consists of two tags, a start tag and an end tag, possibly surrounding text and other elements. The start tag consists of a name surrounded by angle brackets, like ‘<step>’; the end tag consists of the same name surrounded by angle brackets, but with a forward slash preceding the name, like ‘</step>’. The element’s content is everything that appears between the start tag and the end tag, including text and other (child) elements. The following is a complete XML element, with start tag, text content, and end tag:

<step>Knead again, place in a tin, and then bake in the oven.</step>

In addition to content, an element can contain attributes — name-value pairs included in the start tag after the element name. Attribute values must always be quoted, using single or double quotes, and each attribute name should appear only once in any element.

<ingredient amount="3" unit="cups">Flour</ingredient>

In this example, the ingredient element has two attributes: amount, having value ‘3’, and units, having value ‘cups’. In both cases, at the markup level, the names and values of the attributes, just like the names and content of the elements, are just textual data — the ‘3’ and ‘cups’ are not a quantity and unit of measure, respectively, but rather are just character sequences that the document author may be using to represent those things.

In addition to text, elements may contain other elements:

```
<Instructions>
        <step>Mix all ingredients together, and knead thoroughly.</step>
        <step>Cover with a cloth, and leave for one hour in warm room.</step>
        <step>Knead again, place in a tin, and then bake in the oven.</step>
</Instructions>
```

In this case, the Instructions element contains three step elements. XML requires that elements be properly nested — elements may never overlap. For example, this is not well-formed XML, because the em and strong elements overlap:

```
<!-- WRONG! NOT WELL-FORMED XML! -->
<p>Normal <em>emphasized <strong>strong emphasized</em> strong</strong></p>
```

Every XML document must have exactly one top-level root element (alternatively called a document element), so the following would also be a malformed XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- WRONG! NOT WELL-FORMED XML! -->
<thing>Thing one</thing>
<thing>Thing two</thing>
```

XML provides special syntax for representing an element with empty content. Instead of writing a start tag followed immediately by an end tag, a document may contain the empty element tag where a slash follows the element name. The following two examples are functionally equivalent:

```
<foo></foo>
<foo/>
```

There are many more rules necessary to be sure of writing well-formed XML documents, such as the exact characters allowed in an XML name, but this quick tour provides the basics necessary to read and understand many XML documents.

A2.2 SHORT INTRODUCTION ON XML SCHEMA

An XML Schema Definition (XSD) is an instance of an XML schema written in the W3C's XML Schema language. An XSD defines a type of XML document in terms of constraints upon what elements and attributes may appear, their relationship to each other, what types of data may be in them, and other things. It can be used with validation software in order to ascertain whether a particular XML document is of that type.

XSDs were the first W3C-recommended XML schemas to provide a namespace- and datatype-aware alternative to using XML's built-in Document Type Definitions (DTDs). XML Schema Definition files usually have the filename extension '.xsd'.

A2.3 SUMMARY

XTCE is an XML Schema and therefore the file containing XTCE has the '.xsd' extension. The files that will be created with the help of XTCE will be XML files (with '.xml' extension), and once validated against XTCE Schema, they will be XTCE compliant Figure A-1 shows the relations between XML Schema, XTCE Schema and XML Files.

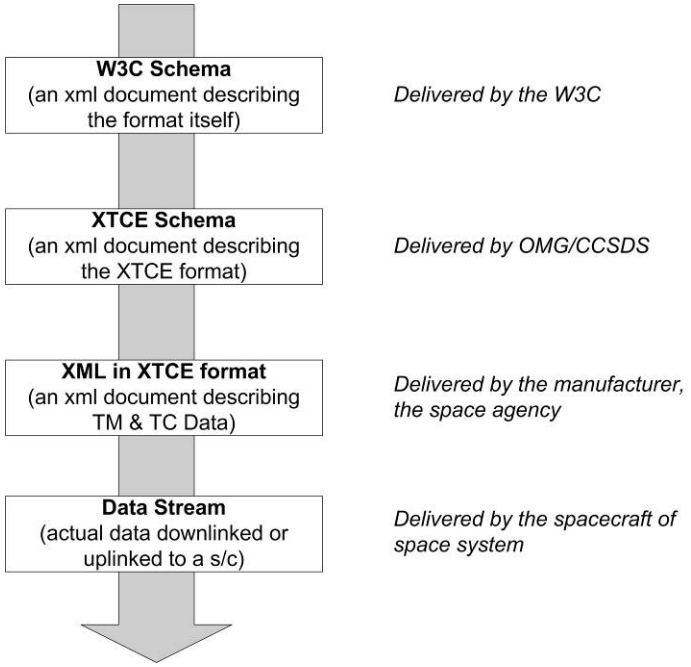

Figure A-1: XML Schema, XTCE Schema, and XML Files

ANNEX B

GLOSSARY

B1 DEFINITION OF TERMS

Here are definitions for specific terms used in this document and their meaning in the XTCE context:

Argument An argument is a value that is part of a telecommand packet. This value is in general set by a human actor, usually a spacecraft controller. Arguments are only used in the context of commanding, never for telemetry purposes.

Command A command is an instruction for a remote system. Spacecraft commands definition usually implies information about coding and packaging the command, validation and verification checks, as well as command execution control.

Container A container is a high level concept that allows describing a package of data. Examples include frames, packets, and minor frames for example.

Element An XML documents it composed of a tree of element. An element is a named tag, which can have associated attributes or not. A tag starts with a '<' and end with a '>'. An element can be empty or can container text of other elements. The tags defining an element come by pairs.
Example: <element attribute="I am an attribute"> some text </element>

List A list is an ordered collection of elements. For example, an argument list is an ordered collection of arguments.

Meta A meta is a prefix used for descriptions. Added to a name, meta declares that the content is a high level description of the name, not the name itself. For example, a Metacommand is a the description of the command, not the command itself.

Parameter A parameter is a value received or set by a machine. Parameters can be telemetered, derived or constant. A meta is a prefix used for descriptions. Added to a name, meta declares that the content is a high level description of the name, not the name itself. For example, a Metacommand is a description of the command, not the command itself.

Ref A ref is a named reference to an element defined elsewhere in the XML document. For example an ArgumentRef is a named reference to an Argument defined elsewhere.

Set A set is an unordered collection of elements. For example, a MetaCommandSet is an unordered collection of command descriptions.

Telemetry Measurement with the aid of intermediate means that permit the measurement to be interpreted at a distance from the primary detector. All measurements on board the spacecraft are transmitted to the ground system in a telemetry stream. Most telemetry measurements will require engineering unit conversion and measurements will have associated validation ranges or lists of acceptable values.

B2 DEFINITION OF ACRONYMS

List of acronyms used in this document:

AIV	Assembly, Integration Verification
CCSDS	Consultative Committee for Space Data Systems
DB	Database
DTD	Document Type Defintions
EGSE	Electrical Ground System Equipment
FDS	Flight Dynamics System
MCS	Mission Control System
MOIMS	Mission Operations and Information Management Systems
MF	Minor Frame
MPS	Mission Planning System
MSB	Most Significant Bit
ODB	Operational Database
OMG	Object Management Group
PGS	Procedure Generation System
S/C	Spacecraft
SDB	Source Database
SDTF	Space Domain Task Force
SIM	Spacecraft Simulator
SM&C	Spacecraft Monitoring and Control
T&C	Telemetry and Command
TDM	Time Division Multiplexing
TC	Telecommand
TM	Telemetric
VC	Virtual Channel
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML Schema Definition
XTCE	XML Telemetric and Command Exchange