## DOCUMENT STATUS SHEET

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue</td>
<td>Date</td>
<td>Change description</td>
</tr>
<tr>
<td>0.1</td>
<td>15 October 2002</td>
<td>Initial version for internal review</td>
</tr>
<tr>
<td>0.2</td>
<td>20 October 2002</td>
<td>Second internal review</td>
</tr>
<tr>
<td>0.3</td>
<td>24 October 2002</td>
<td>Draft</td>
</tr>
<tr>
<td>0.4</td>
<td>16 December 2002</td>
<td>Candidate Draft Submission</td>
</tr>
<tr>
<td>1.0</td>
<td>6 January 2003</td>
<td>Revised Submission</td>
</tr>
<tr>
<td>1.1</td>
<td>28 January 2003</td>
<td>Revised Submission (with documentation corrections)</td>
</tr>
<tr>
<td>1.2</td>
<td>3 March 2003</td>
<td>Updated compliance section and schemas</td>
</tr>
<tr>
<td>1.2</td>
<td>26 March 2003</td>
<td>Revisions requested by Architecture Board</td>
</tr>
</tbody>
</table>
# Change Record Sheet

<table>
<thead>
<tr>
<th>Date</th>
<th>Issue</th>
<th>Section</th>
<th>Description</th>
<th>Reason</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

## 1. PART I

1.1. Copyright Waiver

1.2. Submission Contact Point

1.3. Proposal Overview

1.3.1. Introduction

1.3.2. Background

1.3.3. Introduction

1.3.4. Scope

1.3.5. Summary of Proposed Specification

1.4. Applicable and Reference Documents

1.5. Statement of proof of concept

1.6. Resolution of RFP mandatory and optional requirements

1.7. Glossary of Acronyms

## 2. PART II

2.1. Proposed Specification

2.1.1. The Root Object - The SpaceSystem

2.1.2. The Header Record

2.1.3. Parameters

2.1.4. Command Definitions

2.1.5. Algorithm

2.1.6. Stream

2.1.7. Packaging

2.1.8. Service

2.1.9. Property

2.1.10. NonStandardData

2.1.11. DefaultBusAttributes

2.1.12. DefaultCommandDefinition

2.2. Processing the Telemetry Stream

2.2.1. Typical Delivery Mechanisms for Telemetry Streams

2.2.2. Telecommanding

## 3. PART III

3.1. Introduction/Usage

3.2. Summary of Optional Versus Mandatory Interfaces

3.3. Proposed Compliance Points

3.4. Changes or extensions required to adopted OMG specifications

3.5. Schema Overview

3.6. Explanation of Notation in XMLSpy Diagrams

3.6.1. Element Symbols

3.7. 3.7 Top-level UML Diagrams
3.8. XML Samples
3.9. Complete Schema Definition

Appendix A – XML Schema

SpaceSystem.xsd
Parameter.xsd
CommandDefinition.xsd
Algorithm.xsd
StreamDefinitions.xsd
Packaging.xsd
CommonTypes.xsd

1. 0. PART I

1.1. Copyright Waiver

Lockheed Martin, The Boeing Company, and The European Space Agency waive copyright on this document to the Object Management Group (OMG) and the OMG members for unlimited duplication.

1.2. Submission Contact Point

For enquiries please contact:

Gerry Simon
Lockheed Martin – Mission Systems
Center For Research Support (CERES)
720 Irwin Avenue
Schriever AFB, CO 80912
USA
719-567-8349
gerry.simon@schriever.af.mil

Janice Ann Champion, Senior Staff Engineer
Boeing Satellite Systems
W/EO1/D110
P.O. Box 92919
Los Angeles, CA 90009
(310) 416-4544
janice.a.champion@boeing.com

Mario Merri,
European Space Operations Centre
Robert Bosch Strasse 5,
64293 Darmstadt, Germany
Tel: +49 6151 90 2292
Fax: +49 6151 90 3010
Email: Mario.Merri@esa.int
1.3. Proposal Overview

1.3.1. Introduction

The OMG Space Domain Task Force is pleased to offer this specification to the space industry in the sincere hope that it will bring new levels of interoperability, cost reduction, schedule reduction and risk reduction. This document constitutes a specification by members of the Space Domain Task Force (DTF) of the Object Management Group (OMG), for a Telemetric and Command Data Specification.

1.3.2. Background

Space mission implementations face a very dynamic environment with fast-paced information technology advancement and shrinking space budgets. A more focused use of decreasing public investments in space requires a cost reduction over their entire life cycle, from development up to the end of the useful life of a spacecraft. The use of standards specifications from the early stages of satellite development through mission operation can be utilized to reduce life-cycle cost.

Satellite design and development is performed today through the use of a number of disparate tools and techniques. Interface design to satellite systems and to the payloads the satellites are housing is still a manual and time-consuming effort. Data design, both telemetry and commanding, is still performed multiple times by multiple contractors during the lifecycle of the satellite, well before the satellite is ever deployed for mission operations. The standardization of satellite telemetry and command data for spacecraft health and safety, as well as payload interfaces will reduce the cost of these implementations as well as decrease the schedule of development, integration, and test of the satellite and its component systems. This specification can also be used to support multiple, heterogeneous missions, facilitating interoperability between ground control systems, simulators, testing facilities, etc.

This proposal addresses the need for an information model for telemetry and commanding in support of all phases of the satellite, payload, and ground segment lifecycle: system design, development, test, validation, and mission operations.

1.3.3. Introduction

This is a standard specification to define an information model for spacecraft telemetry and commanding data. This specification is crucial to the general standardization of the telemetry and commanding area.

For a given mission there are a number of lifecycle phases that are supported by a variety of systems and organizations. Additionally, many of these organizations support multiple heterogeneous missions using a common ground segment infrastructure. Telemetry and command definitions must be exchanged among all of these phases, systems, and organizations. This is made difficult and costly because there is no standard method for exchanging this information. The lack of standardization currently requires custom ingestion of the telemetry and commanding information. This customization is inherently error-prone, resulting in the need to revalidate at each step in the lifecycle.

A typical example of this process is between the spacecraft manufacturer and spacecraft-operating agency. The spacecraft manufacturer defines the telemetry and command data in a format that is much different than the one used in the ground segment. This creates the need for database translation, increased testing, software customization, and increased probability of error. Standardization of the
command and telemetry data definition format will streamline the process allowing dissimilar systems to communicate without the need for the development of mission specific database import/export tools. Ideally, a spacecraft operator should be able to transition from one ground system to another by simply moving an already existing command and telemetry database compliant with this command and telemetry database specification.

In addition, standardization will enable space or ground segment simulators to more easily support multiple heterogeneous missions.

As a quick background on spacecraft telemetry and commanding, the following paragraphs will serve to educate those not familiar with spacecraft communications data.

Telemetering is defined (from IEEE Std 1000 [1972]) as “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. Telemetry as used here refers to these measurements whether on-board the spacecraft or transmitted to the ground system. Most telemetry measurements will require engineering unit conversion and measurements will have associated validation ranges or lists of acceptable values.

Commands, as defined for this RFP, are messages originating from the ground or the spacecraft to perform a function on the spacecraft or ground system. Spacecraft commanding usually implies coding and packaging of the command information, validation and verification, as well as authorization to perform. Telemetry and Commanding data are necessarily related to one another, with some command information originating from telemetry and commands relating to particular telemetry measurements. Therefore, the ability to relate individual telemetry with one another and to commands is a very important part of this specification. Packaging of both telemetry and commands can be performed in a number of ways. The most common way to package data for transmission is to use the CCSDS Telemetry and Commanding Packaging format.

**1.3.4. Scope**

The specification addresses the need for a standardized information model capable of supporting TM/TC definitions across the broadest possible range of space domain activities. The goal is to allow TM/TC definitions to be exchanged between different organizations and systems, often at the boundaries of mission phases, without the need for customized import/export, re-validation, or even re-implementation of mission databases.

The scope of this specification is limited to satellite telemetry and commanding data constructs necessary to support satellite and payload data design:
- Telemetry data definitions including support for CCSDS packets as well as TDM frames.
- Data manipulation algorithms to support packaging and unpacking of individual data items.
- Commanding data definitions including command identification, argument specification, and validation criteria.
- Data representation definitions
- Data properties including such things as its default value, validity criteria, and data dependencies.
- The definition of extensible formats such that blocks of information (whether frames of data that are not decommutated or object references or object method calls) can be portrayed in this architecture.

The scope of this specification does not extend to:
• Data distribution mechanisms.
• Command and data protocol specifications.
• RF or analog stream characterization
• Data groupings including aggregation and coherent data sets
• Data Representation (visualization properties)
• Scheduling configuration properties
• Orbital properties

The specification addresses only the definition of TM/TC data, and not the transfer of live or historical TM/TC data.

1.3.5. Summary of Proposed Specification

The proposal has been structured in accordance with the guidelines given in section 4.9 of the RFP, and is divided into 3 main parts:

Part I:
  Copyright Waiver
  Submission Contact Point
  Proposal Overview
  Statement of Proof of Concept
  Resolution of RFP Mandatory and Optional Requirements
  Responses to RFP Issues to be Discussed

Part II:
  Proposed Specification

Part III:
  Summary of Optional versus Mandatory Interfaces
  Proposed Compliance Points
  Changes or Extensions Required to Adopted OMG Specifications [none identified]
  Complete Schema Definition

1.4. Applicable and Reference Documents

All documents published by ESA can be provided on request.

Applicable Documents

| A.01 | CCSDS 102.0-B-4 | CCSDS Packet Telemetry |
| A.02 | CCSDS 203.0-B-1 | CCSDS Telecommand |
| A.03 | ECSS-E-70-41 | Telemetry and Telecommand Packet Utilisation, Draft 5.3, 5 Apr 2001 |
| A.05 | Space/01-04-01 | Telemetric and Command Data Specification, Space RFP-1, 20 Aug 2001 |
1.5.

Statement of proof of concept

The concepts proposed in this document are proven in parts: The first concerns the maturity of the underlying data definitions contained in this proposal. The second concerns its representation in XML.

The database specification on which this proposal is based has been successfully deployed on a several missions over a number of years by each of the companies collaborating in the submission. The specification was first designed in 1994, though its origins can be traced to previous generations of ESA mission control systems (MSSS and SCOS-I). The specification has existed in its current form since 1998. It provides the backbone for ESA’s SCOS-2000 Mission Control System Framework. The base data format/external interface for SCOS-2000 is currently ASCII tab separated files. These files are imported into SCOS-2000’s internal (object oriented) run-time database. For each mission, it has been necessary to write scripts for the population of the database by conversion from an external database, usually provided by the spacecraft manufacturer. In some cases this data is supplemented by data entered manually by spacecraft operations engineers. In other cases conversion is done from a ‘central’ database that is used for both spacecraft tests/checkout and operations.

Boeing Satellite Systems (BSS) has been delivering ground database products to internal and external customer for more than 10 years. The data contents and format has matured over that period. BSS currently delivers data in XML files with DTD files instead of XML Schemas.

Lockheed Martin and its USAF Space Command sponsoring organization SMC Det 12 VO, have over 35 years experience-flying spacecraft from every US spacecraft manufacturer using a variety of custom and COTS ground systems architectures. These spacecraft have served all manner of operational and R&D missions. Lockheed Martin and the USAF fully appreciate the cost, schedule and mission risks associated with adapting existing hardware and software to support new spacecraft and have a keen interest in seeing better industry standardization. The initial schema based Lockheed Martin submission has been operationally employed for over a year.

Reference Documents

| R.01 | ESA PSS-07-101 | Packet Utilisation Standard, Issue 1, May 1994 |
| R.02 | CCSDS 301.0-B-2 | CCSDS Time Code Formats, Issue 2, April 1990 |
| R.03 | http://www.w3.org/TR/REC-xml | W3C Recommendation - Extensible Markup Language (XML) 1.0 (Second Edition, 6 October 2000) |
| R.04 | http://www.w3.org/TR/xmlschema-0/ | W3C Recommendation - XML Schema Part 0: Primer (2 May 2001) |
Space DTF members Boeing, ESA and Lockheed are committed to implementing the XML schema exchange format within 12 months of its final acceptance and look forward to the interoperability benefits it will bring.

1.6. Resolution of RFP mandatory and optional requirements

A statement of compliance to each of the mandatory requirements is given below.

The proposed specification is not compliant to optional requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Compliance</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Telemetry Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Administrative (e.g., telemetry point identifier, telemetry mnemonics (possibly multiple), units)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Categorization (system, subsystem, people, ownership) - multiple values</td>
<td>Part</td>
<td>Specification supports a full system of systems hierarchy; however, telemetry parameters may only “appear” once in the hierarchy.</td>
</tr>
<tr>
<td>Ability to set telemetry items</td>
<td>Yes</td>
<td>Internal 'settable' parameters supported</td>
</tr>
<tr>
<td>Source information (spacecraft address description, &quot;memorized&quot;, derived)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Position dependence (data source reference)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Decommutation support - start byte, start bit, length</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Parsing support - Information is not absolute position dependent (i.e. text message, or event)</td>
<td>Yes</td>
<td>Support for variable packets</td>
</tr>
<tr>
<td>Support for concatenating multiple pieces of source data to form a single telemetry item</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Data Dependence (Telemetry points it relies on, telemetry points that rely on it)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Logical relationships between them</td>
<td>Yes</td>
<td>Validation/Verification/Synthetic expressions</td>
</tr>
<tr>
<td>Identity properties (data type)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Default value</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Representation information (see below for list)</td>
<td>See below</td>
<td></td>
</tr>
<tr>
<td>Algorithmic properties (see below for list)</td>
<td>See below</td>
<td></td>
</tr>
<tr>
<td>Validation checks (red, yellow limits, deltas, possible values - desired and or undesired)</td>
<td>Yes</td>
<td>Alarms</td>
</tr>
<tr>
<td>Validation conditions (when the individual validation checks are used)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><strong>Command Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Administrative (e.g., command identifier, command mnemonic)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Categorization (system, subsystem, people, ownership)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Intrinsic properties (default value)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Fields properties</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Position Dependence (start byte, start bit, length)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Algorithmic Properties (transformation definition)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Data Dependence (Telemetry points it relies on)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Representation (format, precision)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Yes</td>
<td>Group Repeater and Conditional structures</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Support for nesting of fields</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Validation criteria (possible values)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Authorization (who, what, when)</td>
<td>Part</td>
<td>Command Constraints provide a mechanism for &quot;when&quot;, but the submitters all concluded that “who” and “when” are inappropriate for this specification</td>
</tr>
<tr>
<td>Verification information (associated telemetry)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Memorized information (associated mnemonics)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Timing constraints (command dependencies, timing information)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Execution constraints (prompting, not able to execute)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Algorithmic Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Masking and Shifting</td>
<td>Yes</td>
<td>Since telemetry streams are defined in this data spec at the bit level rather than word level, fewer mask and shift operations will be necessary. Masking and shifting operations are created as custom algorithms</td>
</tr>
<tr>
<td>Concatenation</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Polynomial Conversion</td>
<td>Yes</td>
<td>A defined algorithm in the AlgorithmDirectory.xsd. An arbitrary polynomial length is supported.</td>
</tr>
<tr>
<td>Slope Intercept</td>
<td>Yes</td>
<td>Yes. Slope intercept is a special case of a polynomial conversion (i.e. C_0 is the intercept and C_1 is the slope)</td>
</tr>
<tr>
<td>Interpolation</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Table Lookup (and variations)</td>
<td>Yes</td>
<td>Implemented as enumerated</td>
</tr>
<tr>
<td>Enumerated (Discrete Value mapping, i.e. ON/OFF)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Generic (Submitters are also encouraged to provide a generic syntax for algorithmic representation)</td>
<td>Yes</td>
<td>Algorithm inputs, outputs, triggers, and names can be defined as Custom algorithms or algorithm text – in an arbitrary language – may be included. Implementations are not required to support the arbitrary language.</td>
</tr>
<tr>
<td>Identification - (i.e. Mnemonic, Name, Function)</td>
<td>Yes</td>
<td>Algorithm inputs, outputs, triggers, and names can be defined as Custom algorithms.</td>
</tr>
<tr>
<td>Input data</td>
<td>Yes</td>
<td>Algorithm inputs, outputs, triggers, and names can be defined as Custom algorithms.</td>
</tr>
<tr>
<td>Output data</td>
<td>Yes</td>
<td>Algorithm inputs, outputs, triggers, and names can be defined as Custom algorithms.</td>
</tr>
<tr>
<td>Triggering information (data, rate)</td>
<td>Yes</td>
<td>Algorithm inputs, outputs, triggers, and names can be defined as Custom algorithms.</td>
</tr>
<tr>
<td>Equation definition</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Combinations of any of the above techniques</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Data Representation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEEE Float</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>1758 Float</td>
<td>Yes</td>
<td>MIL-STD-1750A floats are a core bus data type.</td>
</tr>
<tr>
<td>Double</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CCSDS Time Formats</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td>Ones Complement Integer</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Twos Complement Integer</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Sign Magnitude Integer</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Bit Order</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Byte Order</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Object References</td>
<td>Partial</td>
<td></td>
</tr>
<tr>
<td>Sequences of the above techniques</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Time tagging of the above data types</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Combinations of any of the above techniques</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

### Validation Techniques

- Equivalency checks: Yes
- Discrete value checks: Yes
- Range checks: Yes
- Delta checks (gradient): Yes
- Trend checks: Yes
- Status checks: Yes
- List checks: Yes
- Parity checks: Yes
- Checksum techniques (XOR at a minimum): Yes
- Combinations of any of the above techniques: Yes

### Packaging Information

- Telemetry Framing information: No
- CCSDS support - packets and frames: Yes
- TDM support - major frame, minor frame: Yes
- Telemetry item support (start byte, start bit, length; combinations of individual fields): Yes
- Command: Yes
- Packaging information - multiple levels of specification: Yes
- Directions for assembling the commands: Yes
- Fields properties: Yes
- Location (start byte, start bit, length): Yes
- Representation (format, precision): Yes
- Nested fields: Yes
- Timing constraints: Yes
- Steam Definition: Yes
- Stream Encoding (e.g., Convolution): Yes
- Encryption: Yes
- Block Encoding (e.g., BCH, CRC): Yes
- Randomization: Yes

### 1.7. Glossary of Acronyms

This section lists the Acronyms used in this document. Note that the acronyms of the tables are defined in the Summary of Tables section. **Review this list before the final submission.**
3. 7. PART II

3.8. Proposed Specification

The proposed specification provides a standard format for defining the Telemetric and Telecommand (TM/TC) data required to perform the processing shown in the figure below. The normative portion of this specification is presented as an XML schema compliant with W3C recommendation of 02/05/2001.
This section presents some background on space systems database concepts to facilitate a better understanding of the schema. The XML schema itself, the subject of this proposal, is described in Part III.

This specification and the XML schemas that makeup the schema is divided into 5 sections: SpaceSystem, CommonTypes, Parameter, CommandDefinition, and Algorithm.

This T&C specification has an object-oriented structure where all the elements of the specification belong to a single root object – the SpaceSystem.

3.8.1. The Root Object - The SpaceSystem
Recognizing that spacecraft operations involve much more than simply controlling the spacecraft, the top-level object is not ‘Spacecraft’ but a more the more generic term ‘SpaceSystem. This name provides
deference to the fact that a spacecraft operations center must control antennas, recorders, ground processing equipment, RF hardware and other many other devices that could potentially share a common T&C data specification; each of these objects is a ‘SpaceSystem’. A SpaceSystem has a Header, zero or more sub-SpaceSystems and a CommandAndTelemetry component. The CommandAndTelemetry component contains the bulk of the Telemetric and Command data and the sub-SpaceSystems give the data a hierarchical structure. The CommandAndTelemetry component contains a ParameterList a CommandDefinition List, an AlgorithmList, a StreamList, a Packaging component, a ServiceList and some optional defaults.

**Note on the sub-SpaceSystem and the hierarchical structure**

Because a SpaceSystem may itself contain other SpaceSystems, the organization of the data may be organized hierarchical structure – similar to the structure of a real space system. The hierarchical organization offers several important advantages over a flat entity list:

- Fewer name space collisions – Almost every spacecraft contains redundant components for reliability or to accomplish the mission. 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 into a unique name. Using a hierarchical namespace, those identical telemetry points can be simply placed into separate sub-SpaceSystems.

- Better organization – modern spacecraft typically have thousands of commands and tens of thousands of telemetry parameters; this number is trending upward. The directory structure provided by this specification provides an improved way to manage this large volume of data. Each subsystem developer can deliver SpaceSystems representing their subsystem without integration issues.

- Defaults at the SpaceSystem level – many of the attributes needed to define spacecraft parameters (e.g. bit order, byte order) are common to most of the parameters in the spacecraft or spacecraft sub-system. This specification allows these attributes to be assigned at the directory level, thereby avoiding their repetition in each parameter.

- Spacecraft which are normally thought of as a SpaceSystem, may actually be sub-SpaceSystems for a constellation of spacecraft SpaceSystems.

- Natural hierarchy – spacecraft designs are increasing in complexity and are normally comprised of systems of systems. The hierarchical organization allowed by a directory structure reflects this.

**Note on Names**

Parameter, and CommandDefinition and other major entity names within this database may be any length but are prohibited from containing the ‘/’, ‘.’, and ‘:’ characters as these are reserved. The ‘/’ is used as the SpaceSystem separator (Unix and HTTP style). The ‘.’ is reserved for future use as a selector for data from other SpaceSystems. The ‘:’ is reserved as an attribute selector.

3.8.2.The Header Record

A SpaceSystem, may contain an optional header record. This record contains some basic context on the data itself (e.g. source, version, revision history, notes, and classification).

3.8.3.Parameters

A Parameter is a description of something that can have a value; it is not the value itself. There are two basic types of Parameters, ‘Parameters’ and a sub-type of Parameter ‘BusParameter’. Parameters are a super class of BusParameters and contain information including data type, description, name, limits, filter
information, and string conversion specifications. BusParameters could have also been called Measurands because they are data points measured off of the Device. In addition to the data contained in Parameter, BusParameters must contain information about bit order, byte order, and extra data type information (may be one of several integer types or floating point types).

Required attributes for Parameters:
- .raw – to refer to the uncalibrated value of a Parameter in its native bit/byte order
- .precalibrated – to refer to the uncalibrated value of a Parameter
- .string – to refer to the string representation of a Parameter

Optional attributes for Parameters:
- .max – the highest seen value of a Parameter
- .min – the lowest seen value of a Parameter

Future attributes for Parameters may include a variety of statistical measures (e.g., average, moving average, standard deviation), temporal measures (e.g., rate-of-change (1st derivative), acceleration (2nd derivative)), and other measures (e.g., samples seen, quality, times out of limit, etc.).

3.8.4. Command Definitions
A Command Definition provides a name the bit representation, and the allowable arguments to create a command that may be sent to a SpaceSystem. The CommandDefinition also contains the instructions for how to perform the binary transformation of this command and its arguments. A CommandDefinition also optionally contains CommandValidations, CommandConstraints, and CommandVerifiers.

A CommandValidation is a process whereby the command issuer is checked as a valid authority to command a device. This check may be temporal and may be command specific. Note: this definition of command definition does not match the traditional one within the spacecraft domain industry; however, this definition does more closely match the definition of validation within the broader IT industry.

A CommandConstraint is a conditional check of the operating environment that allows the Command to proceed.

A CommandVerifier is a conditional check on the telemetry from a SpaceSystem that provides positive indication on the successful execution of a command.

3.8.5. Algorithm
In spacecraft ground systems, it is necessary to perform some specialized processing to process the telemetry, and preprocess commands. There are a number of predefined algorithms and the algorithm section makes it possible to reference externally defined algorithms for arbitrarily sophisticated data processing.

3.8.6. Stream
Spacecraft uplinks and spacecraft downlinks are digital streams of data are there are a number of processing functions that are done on the stream level. The stream section contains the knowledge for how to assemble, disassemble and process spacecraft uplink and downlink streams.
3.8.7. Packaging
The packaging section contains the information required to assemble an uplink from its component parts and disassemble a downlink from its component parts. The packaging section has been created to be extremely generic so that it may be used to define TDM telemetry streams, packetized streams or any other package format.

The packaging schema introduces the major elements: Containers and Messages.

A Container may represent a packet, a frame, a sub-frame or any other grouping/structure of data items. A Container has a sequence of an optional LongDescription element, an ImplementsMessage element, and a Sequence element. The simple form of a Sequence element is an ordered set of Parameter References or other Container References.

A Message holds the identifying bits called a Key and a name of a Container and the location of the Key. There is a MessageList that holds all the Messages for a SpaceSystem. The Key for the Container is held in the Container.ImplementsMessage.Key element. When a Container instance is received, it is identified by a match of its Key instance against the Keys in the MessageList.

3.8.8. Service
A Service is a named collection of messages and containers. An example is an onboard memory dump service that has a set of messages and containers allocated to provide this Service.

3.8.9. Property
A property is a hierarchical set of names and values. These are meant as a placeholder for information that does not otherwise fit into this schema. This data needs to be in the name/value form.

3.8.10. NonStandardData
This is of type xs:any and may also be used to include a block of any arbitrarily formatted XML. This section may be used for any other data required by the ground system or factory that that does not otherwise fit into the schema.

3.8.11. DefaultBusAttributes
All parameters and command fields within a SpaceSystem may contain BusAttributes. BusAttributes contains the bit order, byte order, calibration, and error detection and correction for the parameter. Since these bus attributes are commonly the same for an entire SpaceSystem (or sub-system), they may be defined once at each SpaceSystem level with this element.

3.8.12. DefaultCommandDefinition
This Element defines a default CommandDefinition for the entire SpaceSystem or sub-system.

3.9. Processing the Telemetry Stream
This section describes the processes that deal with Processing the Telemetry Stream using data in the underlying XML schema.
3.9.1. Typical Delivery Mechanisms for Telemetry Streams

Packetized Telemetry

Over the last decade, ESA has been developing and promoting the Packet Utilization Standard (PUS), first through its own PSS standards series [R.01], and latterly through the European Co-operation for Space Standardization (ECSS) [A.03]. The PUS supports the concept of end-to-end services, in which on-board processes communicate directly with peer processes on the ground, through a set of defined message structures. Services typically comprise both downlink [TM] and uplink [TC] messages. The PUS identifies a standard set of services with associated message structures overlaid on the existing Consultative Committee for Space Data Systems (CCSDS) packet TM/TC standards, and also provides a framework for the definition of mission specific services.

Most TM/TC formats support basic housekeeping telemetry and simple device commanding, and while PUS does offer benefits here, it is with other classes of data uplink/downlink that the service concept comes into its own. The identification of standard command verification, on-board memory management and on-board schedule management services, has enabled standardized capabilities to be implemented to support these functions across a range of missions, when previously this had required mission-specific development.

Future ESA spacecraft will be compatible with the PUS, and as a consequence the concepts it espouses are being built into the product lines of most European spacecraft manufacturers. The following space missions have used or plan to use the PUS: XMM, Meteosat Second Generation (MSG), INTEGRAL, GOMOS (Envisat Instrument), ATV (Automated Transfer Vehicle), Orsted (Danish microsatellite), PROBA, ROSETTA, MARS EXPRESS, FIRST/PLANK, CRYOSAT, GOCE, GALILEO.

From a European perspective, it is critical that the Telemetric and Command Data Specification adopted by OMG should provide for the representation of PUS-compliant telemetry and telecommand definitions. A key objective of the specification should be that it fosters interoperability and compatibility of systems within the Space domain, through convergent TM/TC definitions. This will only be achieved by an inclusive specification that takes into account compatibility with existing standards.

ESA's own satellite control system kernel, SCOS-2000, has been designed to support both packet TM/TC and the PUS service concepts. This approach does not, however, limit its applicability to missions using these standards: it has generally proven to be much easier to map other [simpler] TM/TC structures, such as fixed format frame-based telemetry, on to the service/packet model than to do this the other way around. SCOS-2000, although designed as a mission operations support tool, has also been successfully deployed in support of spacecraft integration and test activities.

SCOS-2000 won two awards at the OMG's Object World conference in Berlin, 1999:

1st prize for "Best Implementation of a Distributed Application using Object Technology"

Finalist for "Best use of Object Technology within an Enterprise or Large System Environment".

SCOS-2000 is being used for all recent and future ESA spacecraft missions, including: Huygens, MTP LEOP, Proba, MSG LEOP, Integral, Rosetta, Mars Express, SMART-1, Cryosat and GOCE. It has also been provided to other space agencies, which are using, or plan to use, SCOS in the control of their own missions: these include the German (DLR) and Canadian Space Agencies (CSA) and EUTELSAT.
**Time Division Multiplexing**

A *telemetry stream* can be defined as a long stream of binary data originating on the spacecraft that contains health, status and other information generated by the spacecraft bus and payload. *Decommutation* is defined as the process by which the telemetry stream is broken into component pieces for use by the ground software. This section is devoted to explaining how to use the decommutation records to correctly break apart the frame-based telemetry stream.

**Telemetry Decommutation**

TDM streams are uniquely characterized by the presence of a predictable telemetry format. This is because spacecraft generating a TDM stream stores a table of sequencing code consisting of ordered sets of telemetry requests. Since these sequencing code tables are known on the ground, the order in which telemetry requests are generated on the spacecraft and assembled into the telemetry stream can also be predicted. It is only that telemetry that we can predict which can be decommutated using the method contained in this section. The spacecraft generates a repeating stream of bits and sends it to the ground. Each stream of bits is generated using a set of directions that direct the spacecraft to include specific pieces of data in specific locations within the stream. Each set of directions is called a telemetry *format*. Since multiple formats usually exist, the spacecraft is typically restricted from generating more than one format at any given time.

The database differentiates between distinct sets of decommutation information using a *format number*. Each format number contains one complete set of directions for the decommutation of a designated telemetry stream. In order to determine which format the spacecraft is producing at any given time, a fixed location common to all formats is found in the telemetry stream that is devoted to the identification of the format number.

A *minor frame* is a subset of a format. Each format is typically broken into minor frames. Minor frames are characterized by their repetition of specific telemetry measurands in specific locations. For example, every minor frame begins with information that identifies the format number as well as a minor frame number. Once a minor frame has been identified within a format, it may be broken into its component *words*. Words are composed of some predetermined numbers of individual bits; typical words are eight bits long. The *start bit* represents the first bit within the 8-bit word of the designated minor frame and format. The identification of the format, minor frame, word and start bit uniquely references the starting location for a Parameter. Once this information is declared, the length of the Parameter can be obtained and the binary string from the telemetry stream can be assigned to the Parameter for additional processing. This additional processing typically includes assigning a name, performing EU conversion and limit checking.

The following illustration has been provided to help visualize how a *major frame* is broken into its component parts. A major frame is the repeating section of a format and consists of minor frames, words and start-bits. Major frames will repeat as long as the spacecraft is generating telemetry; major frame numbers have no meaning.
Each format generates a repeating stream of telemetry called a major frame. Each format is broken into some predetermined number of minor frames, usually 32. These minor frames are displayed in a stacked form as shown at the bottom of this page in horizontal rows. Minor frames always have a domain that starts at zero and increments in integer steps. (i.e. 0, 1, 2, 3, …)

Minor frames are additionally broken into some predetermined number of words. This means that a word is a subset of a minor frame. This is because all telemetry requests from the spacecraft consist of single word requests. Words are displayed as vertical columns in the "grid" at the bottom of this page. Words always have a domain that starts at zero and increments in integer steps. (i.e. 0, 1, 2, 3, …)

Finally, each word is broken into some predetermined number of bits, usually 8. Once the word is broken into bits, the reference for Parameter locations can begin. All Parameters will be referenced by format, minor frame, word and start bit. The number of bits to assign to the Parameter is located within the Parameter definition itself. The total number of bits in a major frame can be calculated using the following equation:

\[
\text{#of bits} = \frac{\text{#of minor frames} \times \text{#of words} \times \text{#of bits}}{\text{major frame} \times \text{minor frame} \times \text{word}}
\]

The illustration below has been provided to show how a major frame is typically presented. Notice that minor frame and word numbers start at zero at the origin. This illustration will be useful when the concepts of dwell and compression are introduced later.
Table 1 Sample Normal Telemetry Frame File

<table>
<thead>
<tr>
<th>Parameter</th>
<th>format</th>
<th>minor_frame</th>
<th>word</th>
<th>start_bit</th>
<th>major_frame_rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>30050</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>30051</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>30052</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>30053</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>30054</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>30055</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>30056</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>30057</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>32</td>
</tr>
<tr>
<td>30058</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>30059</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>30060</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>30076</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>31180</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>31220</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>31060</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>31061</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>32010</td>
<td>1</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>30130</td>
<td>1</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>
Table 1 has been filtered for format = 1 records and shows the first twenty records correctly sorted by minor frame, word then start bit. Each column contains information that may not be discarded by any real-time system if it is to correctly interpret the information being received by the spacecraft. Columns 1 through 5 have been discussed already, so the remainder of this section will be on the remaining columns: major frame rate and non-preemptable.

**Dwell**
Dwell is a special TDM mode in which portions of the telemetry stream is preempted by customized telemetry requests in specified word locations. This document will not detail all of the unique requirements associated with the dwell function, instead it will explain the interfaces between the dwell function and the data tables.

**Configuring for Dwell**
Before the telemetry stream will be preempted with dwell telemetry, the operator will configure for dwell using commands to identify which specific telemetry requests will occur in which words of the telemetry stream. The spacecraft must then be commanded to switch from normal mode into dwell mode before the telemetry stream will actually change. Note that telemetry modes are still dependent upon the format number. The commands to configure for dwell and to switch modes are contained within the database.

**Preemptability**
Decommutation records contain all of the directions for decommutating the normal-mode (or normal) telemetry stream. When the spacecraft switches into dwell mode, portions of the normal telemetry frame get over-written with dynamic records wherever the record is labeled as preemptable.

3.9.2.**Telecommanding**

**Command Construction**
Directions for the assembling all bits of the common command are stored in the database. The database uses an argument driven structure and to avoid storing repeating values default arguments may be defined at any point in the heirarchical spacecraft level. Specifying this option each time for the thousands of commands is unnecessary and introduces the possibility of inconsistencies within the data.

**Command Assembly**

**Variable (v)**
- A field type of variable shall be used when the data type of the field depends upon the address of the parameter being loaded. For example, the SCP Memory Upload Command (ID AE) allows the upload of up to 50 16-bit words that start at an operator specified address. The data type of each of these 50 words depends upon each of the 50 addresses. The data type for these variable type fields shall be limited to the following: integer, long integer, 1750A float, 1750A long float, IEEE float, and IEEE long float. These fields will have an associated sign flag for data encoding. They will require operator input and the operator’s value must be within the associated minimum value and maximum value. Variable fields are associated with multiword commands since they require four 16-bit fields to transmit the value to the spacecraft.

**Odd Parity (p)**
A field type of odd parity shall be used for fields whose value must be calculated as “0” or “1” as the command is constructed in order to yield a command with odd parity. This field type requires calculation by the ground software since it typically cannot be calculated in advance and should not be assigned to the operator for determination.

Even Parity (q)
A field type of even parity shall be used for fields whose value must be calculated as “0” or “1” as the command is constructed in order to yield a command with even parity. This field type requires calculation by the ground software since it typically cannot be calculated in advance and should not be assigned to the operator for determination.

5. PART III

5.1. Introduction/Usage
The Telemetry & Command data specification is intended as a way to describe telemetry and command “databases” as used in ground telemetry systems, packet and TDM based systems. The intent is to allow the easy interchange of these databases between systems and organizations.

The vision is that this standard will one day be the “native” format for ground systems. Until that time, companies and organizations using this specification can employ converters to go from one system to another, or can convert an existing database into this format for exchange with other parties.

5.2. Summary of Optional Versus Mandatory Interfaces
Being a data standard, this standard does not specify any interface.

5.3. Proposed Compliance Points
The xsd files in Appendix A are normative. A compliant database is an XML file that complies with these xsd files. Fully compliant implementing software will interpret and/or generate any databases compliant with this specification. Compliant implementing software will interpret and/or generate all database elements required by the schema.

5.4. Changes or extensions required to adopted OMG specifications
None.

5.5. Schema Overview
The Schema is organized into seven separate W3C schema files: SpaceSystem.xsd, Parameter.xsd, CommonTypes.xsd, Packaging.xsd, Algorithm.xsd, StreamDefinitions.xsd, and CommandDefinitions.xsd.

SpaceSystem.xsd contains the schema for a SpaceSystem. A SpaceSystem is a collection of SpaceSystem(s) including space assets, ground assets, multi-satellite systems and sub-systems. A SpaceSystem is the root element for the set of data necessary to monitor and command an arbitrary space
device - this includes the binary decomposition the data streams going into and out of a device. The major data component of a Space System is the TelemetryAndCommanding section where all Telemetric and Command data is stored. Additionally, a Space System has a Header record, containing general information on the data, and 'Algorithms' containing certain well known algorithms and references to external algorithms for mission unique processing.

Parameter.xsd contains the schema for a Parameter. A Parameter is a description of something (usually from telemetry) that can have a value; it is not the value itself. Parameters have a data type, and may have native data type attributes (e.g. integer format). Parameters may also have alarms, validity checks, descriptions and data on how to represent the value of the parameter as a string.

CommonTypes.xsd – Is a collection of W3C schema data types that are re-used throughout this schema specification.

Packaging.xsd – Contains the dictionary for messages and containers, which in turn describe the physical composition of data in a communication system.

Algorithm.xsd – Contains the structure for an Algorithm. An Algorithm may be one of a growing set of pre-defined algorithms or a named escape into a user defined algorithm where (depending on the system) the name of the algorithm may be a java class, a function in a shared library, an external program or some other reference to an outside algorithm.

StreamDefinitions.xsd - provides a language for defining binary stream data.

CommandDefinition.xsd - defines the structure for a CommandDefinition. A CommandDefinition provides a description of the command, a name for the command, the allowable arguments for the command, and (optionally) for binary commands destined for an off-platform bus) the opcode for the command and the binary transformation of the command.

5.6. Explanation of Notation in XMLSpy Diagrams

5.6.1. Element Symbols

Alias

The above symbol is a mandatory single element.

LongDescription

A mandatory single element, containing Parsed Character Data.
Here the element Header is followed by a sequence of three optional elements. A sequence requires order of element is maintained and is represented by the icon with three dots. Optional elements are shown by a dashed outline. The range of occurrences of each element is shown by the numbers under the element icon. Each of these three elements can occur 0 to infinity times. When no range of occurrence is shown then one and only one occurrence is assumed.
Below is an example XMLSpy diagram with all the other major XMLSpy graphical elements annotated with their meaning.

5.7. 3.7 Top-level UML Diagrams

Key UML diagrams of the SpaceSystem schema are included for reference on the following pages.
Figure 1 - Top Level SpaceSystem Element

Figure 2 - TelemetryAndCommanding.Packaging
Figure 3 - TelemetryAndCommanding.CommandDefinitionList.CommandDefinition
Figure 4 - TelemetryAndCommanding.ParameterList.Parameter

![Diagram showing AlgorithmList with CustomAlgorithm, ConcatenationAlgorithm, MathAlgorithm, ReedSolomonEncoder, and ReedSolomonDecoder]

Figure 5 - TelemetryAndCommanding.AlgorithmList
Figure 6 - TelemetryAndCommanding.AlgorithmList..CustomAlgorithm
Figure 7 - TelemetryAndCommanding.AlgorithmList.MathAlgorithm

Property
name : NameReferenceType
shortDescription [0..1] : string
longDescription [0..1] : string
value : string

Property

Figure 8 - TelemetryAndCommanding.PropertyList.Property

Service
name : ServiceNameType
shortDescription [0..1] : ShortDescriptionType
longDescription [0..1] : LongDescriptionType

MessageList  ContainerList

Figure 9 - TelemetryAndCommanding.ServiceList.Service

MessageList
NameRef [1..*] : NameReferenceType

ContainerList
Id [1..*] : ContainerNameType
5.8. **XML Samples**

There are several sample XML documents available in the zip file space/2003-03-10. The XML file “PUS Packets.xml” is a short sample document containing a CCSDS/ ESA PUS spacecraft. The XML file “Boeing TDM.xml” is a short sample document containing a TDM Boeing type spacecraft. The XML file “SimpleSat.xml” is a short sample containing a very simple satellite, but demonstrates some of the more arcane features of the schema.

5.9. **Complete Schema Definition**

This section contains the Telemetry and Commanding schema. The Schemas have been validated and checked for well-formedness using the XML Spy 4.4.

The Schemas are presented as in a graphical form as output by XMLSpy. The XML Schema itself is in Appendix A.

**Schema SpaceSystem.xsd**

<table>
<thead>
<tr>
<th>Schema location</th>
<th>targetNamespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpaceSystem.xsd</td>
<td><a href="http://www.omg.org/space/tcspec">http://www.omg.org/space/tcspec</a></td>
</tr>
</tbody>
</table>

**Elements**

<table>
<thead>
<tr>
<th>Complex types</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpaceSystem</td>
</tr>
</tbody>
</table>

**Complex types**

<table>
<thead>
<tr>
<th>Elements</th>
<th>Complex types</th>
</tr>
</thead>
<tbody>
<tr>
<td>FixedFrameSync</td>
<td>CalibratorType</td>
</tr>
<tr>
<td></td>
<td>ConcatenationAlgorithmType</td>
</tr>
<tr>
<td></td>
<td>CustomAlgorithmType</td>
</tr>
<tr>
<td></td>
<td>MathAlgorithmType</td>
</tr>
<tr>
<td></td>
<td>ReedSolomonType</td>
</tr>
<tr>
<td></td>
<td>TriggerType</td>
</tr>
</tbody>
</table>

**Elements**

<table>
<thead>
<tr>
<th>Complex types</th>
</tr>
</thead>
<tbody>
<tr>
<td>DwellSet</td>
</tr>
</tbody>
</table>

**Complex types**

<table>
<thead>
<tr>
<th>Elements</th>
<th>Complex types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ComplexAlarmsType</td>
</tr>
<tr>
<td></td>
<td>OffsetParameterRefType</td>
</tr>
<tr>
<td></td>
<td>ParameterRefMatchListType</td>
</tr>
<tr>
<td></td>
<td>ParameterRefType</td>
</tr>
</tbody>
</table>
ParameterType  
PhysicalAddressType  
RangeListType  
SimpleAlarmsType  
StopType  
VariableParameterType

schema location:  CommonTypes.xsd  
targetNamespace:  http://www.omg.org/space/tcspec

Complex types  
AbsoluteTimeType  
AliasType  
BaseDataType  
BaseParameterType  
BitOffsetType  
BusAttributesType  
ComparisonCheckType  
DefaultValueType  
HeaderType  
IntegerRangeType  
MathOperationType  
NonStandardDataType  
OccursType  
PolynomialType  
PropertyType  
RangeType  
RealRangeType  
SimpleParameterRefType  
SplinePointType  
SystemNameType  
ToStringType  
UnitType

Complex types  
ArgumentType  
CommandConstraintType  
CommandDefinitionType  
CommandVerifierType

schema location:  CommandDefinition.xsd  
targetNamespace:  http://www.omg.org/space/tcspec

Complex types  
ArgumentType  
CommandConstraintType  
CommandDefinitionType  
CommandVerifierType  
InterlockScopeType  
StageType

schema location:  Packaging.xsd  
targetNamespace:  http://www.omg.org/space/tcspec

Elements  
Packaging
Complex types  
ChoiceType  
ContainerRefType  
ContainerType  
DynamicContainerRefType

Simple types  
TypeName  
MessageKeyIdType  
MessageNameType  
ServiceNameType
MessageKeyMatchListType
MessageKeyMatchType
MessageKeyType
MessageType
PackagingDefinitionType
SequenceType
ServiceType

Complex types
ConvolutionalStreamType
EncryptedStreamType
FixedFrameStreamType
PCMStreamType
StreamType
SyncStrategyType
VariableFrameStreamType

Simple types
StreamSource.Type

schema location: StreamDefinitions.xsd
targetNamespace: http://www.omg.org/space/tcspec
element SpaceSystem

diagram

cmplxType SpaceSystemType

namespace http://www.omg.org/space/tcspec
tc:SpaceSystemType

Header
LongDescription

0..∞

tc:SpaceSystem

TelemetryAndCommanding

NonStandardData

TelemeetryAndCommanding is the root element for the set of data necessary to monitor and command an arbitrary space device - this includes the binary decomposition the data streams going into and out of a device. TelemetryAndCommanding is composed of the major sub-components 'Parameters' and 'CommandDefinitions'. Additionally, TelemetryAndCommanding has a Header record, containing general information on the data, and 'Algorithms' containing certain well known algorithms and references to external algorithms for mission unique processing.

children

Header
LongDescription
tc:SpaceSystem
TelemetryAndCommanding
NonStandardData

used by attributes

name string
shortDescription tc:ShortDescriptionType

complexType SpaceSystemType

name                   Type                                  Use                      Default                       Fixed              Annotation

namespace type
tc:SpaceSystemType

Header LongDescription tc:SpaceSystem TelemetryAndCommanding NonStandardData

complexType SpaceSystemType

name string
shortDescription tc:ShortDescriptionType
SpaceSystem is a collection of SpaceSystem(s) including space assets, ground assets, multi-satellite systems and sub-systems. A SpaceSystem is the root element for the set of data necessary to monitor and command an arbitrary space device - this includes the binary decomposition the data streams going into and out of a device. The major data component of a SpaceSystem is the TelemetryAndCommanding section where all Telemetric and Command data is stored. Additionally, a SpaceSystem has a Header record, containing general information on the data, and 'Algorithms' containing certain well known algorithms and references to external algorithms for mission unique processing.
element SpaceSystemType/LongDescription

diagram

```
namespace http://www.omg.org/space/tcspec

tc:HeaderType

Author 0..∞

Note 0..∞

History 0..∞

```

<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>classification</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unclassified

element SpaceSystemType/TelemetryAndCommanding
TelemetryAndCommanding is the root element for the set of data necessary to monitor and command an arbitrary space device - this includes the binary decomposition the data streams going into and out of a device. TelemetryAndCommanding is composed of the major sub-components 'Parameters' and 'CommandDefinitions'. Additionally, TelemetryAndCommanding has a Header record, containing general information on the data, and 'Algorithms' containing certain well known algorithms and references to external algorithms for mission unique processing.

namespace http://www.omg.org/space/tcspec
children DefaultBusAttributes DefaultCommandDefinition CommandDefinitionList ParameterList tc:Packaging PropertyList ServiceList AlgorithmList StreamList NonStandardData
annotation documentation TelemetryAndCommanding is the root element for the set of data necessary to monitor and command an arbitrary space device - this includes the binary decomposition the data streams going into and out of a device. TelemetryAndCommanding is composed of the major sub-components 'Parameters' and 'CommandDefinitions'. Additionally, TelemetryAndCommanding has a Header record, containing general information on the data, and 'Algorithms' containing certain well known algorithms and references to external algorithms for mission unique processing.

element SpaceSystemType/TelemetryAndCommanding/DefaultBusAttributes
namespace http://www.omg.org/space/tcspec
tc:BusAttributesType
ErrorDetectCorrect DefaultCalibrator ByteOrder ContextCalibrator

<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bitLength</td>
<td>unsignedInt</td>
<td>required</td>
<td></td>
<td></td>
<td>MostSignificantBitFirst</td>
</tr>
<tr>
<td></td>
<td>bitOrder</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>busType</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

element SpaceSystemType/TelemetryAndCommanding/DefaultCommandDefinition
Diagram:

- **LongDescription**: 
  - **Alias**: 0..∞
  - **SystemName**: 
    - **Interlock**: Specifies the interlock for this command, including the scope of the interlock. An interlock restricts commanding ability at the specified scope until the command has reached the specified state.
    - **Field**: 0..∞
  - **BinaryTransform**: 0..∞
  - **CommandProcessor**: A command processor is the name of some external algorithm that is used to format the command. Command processors may also perform additional formatting of the command - formatting that cannot be adequately described in the database.
  - **CommandValidator**: 0..∞
    - **CommandVerifier**: A command validator is used to validate that a particular user has the proper permissions to issue the command. Command Validators - like command processors - are the name of some external algorithm that does the command validation.
    - **CommandConstraint**: 
      - **CommandVerifier**: 
        - **ParametersToSet**: 0..∞
          - **NonStandardData**: Set the values of parameters after the command has been verified.
<table>
<thead>
<tr>
<th>namespace</th>
<th><a href="http://www.omg.org/space/tcspec">http://www.omg.org/space/tcspec</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td><code>tc:CommandDefinitionType</code></td>
</tr>
<tr>
<td>children</td>
<td><code>LongDescription</code> <code>Alias</code> <code>SystemName</code> <code>Interlock</code> <code>Field</code> <code>BinaryTransform</code> <code>CommandProcessor</code> <code>CommandValidator</code> <code>CommandConstraint</code> <code>CommandVerifier</code> <code>ParametersToSet</code> <code>NonStandardData</code></td>
</tr>
<tr>
<td>attributes</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td><code>tc:NameType</code></td>
</tr>
<tr>
<td>type</td>
<td>required</td>
</tr>
<tr>
<td>use</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td></td>
</tr>
<tr>
<td>fixed</td>
<td></td>
</tr>
<tr>
<td>annotation</td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td><code>tc:ShortDescriptionType</code></td>
</tr>
<tr>
<td>critical</td>
<td><code>boolean</code></td>
</tr>
<tr>
<td></td>
<td>optional</td>
</tr>
<tr>
<td>highPriority</td>
<td><code>boolean</code></td>
</tr>
<tr>
<td></td>
<td>optional</td>
</tr>
<tr>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>
element SpaceSystemType/TelemetryAndCommanding/CommandDefinitionList

```
<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td><a href="http://www.omg.org/space/tcspec">http://www.omg.org/space/tcspec</a></td>
</tr>
<tr>
<td>children</td>
<td>ArgumentDefinition CommandDefinition</td>
</tr>
<tr>
<td>identity</td>
<td>key</td>
</tr>
<tr>
<td>constraints</td>
<td>documentation</td>
</tr>
<tr>
<td>annotation</td>
<td>Hold the list of CommandDefinitions</td>
</tr>
</tbody>
</table>
```

element SpaceSystemType/TelemetryAndCommanding/CommandDefinitionList/ArgumentDefinition
element            SpaceSystemType/TelemetryAndCommanding/CommandDefinitionList/
CommandDefinition
namespace http://www.omg.org/space/tcspec
tc:CommandDefinitionType

children:
- LongDescription
- Alias
- SystemName
- Interlock
- Field
- BinaryTransform
- CommandProcessor
- CommandValidator
- CommandConstraint
- CommandVerifier
- ParametersToSet
- NonStandardData

attributes:

Name | Type | Use | Default | Fixed | Annotation
--- | --- | --- | --- | --- | ---
name | tc:NameType | required | | | |
shortDescription | tc:ShortDescriptionType | optional | | | |
critical | boolean | optional | false | | |
highPriority | boolean | optional | false | | |
element **SpaceSystemType/TelemetryAndCommanding/ParameterList**

**Diagram**

```
ParameterList --- Parameter

ParameterList Holds the list of parameter definitions
Parameter A single parameter definition
```

**Namespace**

`http://www.omg.org/space/tcspec`

**Children**

- `Parameter`

**Identity Constraints**

- `key` ParameterListKey

**Annotation Documentation**

Holds the list of parameter definitions

**Element** **SpaceSystemType/TelemetryAndCommanding/ParameterList/Parameter**
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>tc:NameType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dataSource</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>readOnly</td>
<td>boolean</td>
<td>optional</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>validRangeApplies</td>
<td>boolean</td>
<td>optional</td>
<td>true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ToCalibrated</td>
<td>boolean</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>onboardID</td>
<td>string</td>
<td>optional</td>
<td></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>documentation</td>
<td>A single parameter definition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
element `SpaceSystemType/TelemetryAndCommanding/PropertyList`

```
namespace http://www.omg.org/space/tcspec
children Property
```

element `SpaceSystemType/TelemetryAndCommanding/PropertyList/Property`

```
namespace http://www.omg.org/space/tcspec
tc:PropertyType
children Property
attributes Name                   Type                                  Use                      Default                       Fixed              Annotation
name                    tc:NameReferenceType     required
shortDescription    string
longDescription    string
value                    string                                  required
annotation
documentation  Using these properties is discouraged, they are only meant as a placeholder for date that does not otherwise fit into this schema.
```

element `SpaceSystemType/TelemetryAndCommanding/ServiceList`

```
namespace http://www.omg.org/space/tcspec
children Service
attributes key                                 Selector                           Field(s)
name ServiceListKey                          ./Service                          @name
```

element `SpaceSystemType/TelemetryAndCommanding/ServiceList/Service`

```
```
namespace http://www.omg.org/space/tcspec
tc:ServiceType

LongDescription MessageList ContainerList

attributes
Name                   Type                                  Use                      Default                       Fixed              Annotation
ame                    tc:ServiceNameType         required                                                                     documentation
shortDescription    tc:ShortDescriptionType    optional                                                                     documentation

Optional long description of this service
Set of messages this service uses
Set of containers this service uses

Unique identifier for this service set
Optional short description of this service

element SpaceSystemType/TelemetryAndCommanding/AlgorithmList

namespace http://www.omg.org/space/tcspec

tc:AlgorithmList

children
CustomAlgorithm ConcatenationAlgorithm MathAlgorithm ReedSolomonEncoder ReedSolomonDecoder

Optional short description of this service

1..*
element SpaceSystemType/TelemetryAndCommanding/AlgorithmList/ConcatenationAlgorithm
element `SpaceSystemType/TelemetryAndCommanding/AlgorithmList/MathAlgorithm`
namespace http://www.omg.org/space/tcspec
tc:StreamType
tc:NonStandardDataType

element SpaceSystemType/TelemetryAndCommanding/NonStandardData

diagram

namespace http://www.omg.org/space/tcspec
tc:NonStandardDataType
tc:FixedFrameSync
An algorithm most often implemented in hardware, that finds the frame synchronization mark in each frame. Given raw binary data, a PCM Fixed Frame Sync will output framed data. Triggers on the input parameter.

element tc:FixedFrameSync/InputParameterRef

element tc:FixedFrameSync/OutputParameterRef
<table>
<thead>
<tr>
<th>children attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>parameterName</td>
<td>tc:NameReferenceType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>spaceSystemName</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
element tc:FixedFrameSync/FixedFrameStream

diagram

complexType tc:CalibratorType

diagram

namespace http://www.omg.org/space/tcspec

SplineCalibrator PolynomialCalibrator ToStringCalibrator
elements tc:BusAttributesType/ContextCalibrator/Calibrator tc:BusAttributesType/DefaultCalibrator

name string optional
shortDescription tc:ShortDescriptionType optional
documentation Calibrators are normally used to convert to and from bit compacted numerical data
element `tc:CalibratorType/SplineCalibrator`

A calibration type where a segmented line in a raw vs calibrated plane is described using a set of points. Raw values are converted to calibrated values by finding a position on the line corresponding to the raw value. The algorithm triggers on the input parameter.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>positiveInteger</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extrapolate</td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```

```
annotation documentation  A calibration type where a segmented line in a raw vs calibrated plane is described using a set of points. Raw values are converted to calibrated values by finding a position on the line corresponding to the raw value. The algorithm triggers on the input parameter.
```

element `tc:CalibratorType/SplineCalibrator/SplinePoint`

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>positiveInteger</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>raw</td>
<td>double</td>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>calibrated</td>
<td>double</td>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```

element `tc:CalibratorType/PolynomialCalibrator`

A calibration type where a curved in a raw vs calibrated plane is described using a set of polynomial coefficients. Raw values are converted to calibrated values by finding a position on the curve corresponding to the raw value. The first coefficient belongs with the X^0 term, the next coefficient belongs to the X^1 term and so on. The algorithm triggers on the condition parameter.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>positiveInteger</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
A calibration type where a curved in a raw vs calibrated plane is described using a set of polynomial coefficients. Raw values are converted to calibrated values by finding a position on the curve corresponding to the raw value. The first coefficient belongs with the $X^0$ term, the next coefficient belongs to the $X^1$ term and so on. The algorithm triggers on the condition parameter.
element `tc:CalibratorType/ToStringCalibrator`

A calibration type where the bus value is converted to a string value.

complexType `tc:ConcatenationAlgorithmType`

An algorithm that will concatenate two or more parameters. Although not checked by the schema, input parameters must be binary or boolean and the resulting output parameter must be sized to contain the concatenation.

documentation

```
A calibration type where the bus value is converted to a string value.

complexType `tc:ConcatenationAlgorithmType`

An algorithm that will concatenate two or more parameters. Although not checked by the schema, input parameters must be binary or boolean and the resulting output parameter must be sized to contain the concatenation.

```

children `InputParameterList` `OutputParameterRef`

namespace `http://www.omg.org/space/tcspec`

used by `SpaceSystemType/TelemetryAndCommanding/AlgorithmList/ConcatenationAlgorithm`

annotation
documentation

```
A calibration type where the bus value is converted to a string value.

complexType `tc:ConcatenationAlgorithmType`

An algorithm that will concatenate two or more parameters. Although not checked by the schema, input parameters must be binary or boolean and the resulting output parameter must be sized to contain the concatenation.

```

children `InputParameterList` `OutputParameterRef`

```
A calibration type where the bus value is converted to a string value.

complexType `tc:ConcatenationAlgorithmType`

An algorithm that will concatenate two or more parameters. Although not checked by the schema, input parameters must be binary or boolean and the resulting output parameter must be sized to contain the concatenation.

```

children `InputParameterList` `OutputParameterRef`

```
A calibration type where the bus value is converted to a string value.

complexType `tc:ConcatenationAlgorithmType`

An algorithm that will concatenate two or more parameters. Although not checked by the schema, input parameters must be binary or boolean and the resulting output parameter must be sized to contain the concatenation.

```

children `InputParameterList` `OutputParameterRef`

```
A calibration type where the bus value is converted to a string value.

complexType `tc:ConcatenationAlgorithmType`

An algorithm that will concatenate two or more parameters. Although not checked by the schema, input parameters must be binary or boolean and the resulting output parameter must be sized to contain the concatenation.

```

children `InputParameterList` `OutputParameterRef`
element **tc:ConcatenationAlgorithmType/OutputParameterRef**

complexType **tc:CustomAlgorithmType**
The type definition used by the Custom Algorithm element.

**LongDescription**

**Trigger**

**Inputs**

**OutputParameterRef**

Names an output parameter to the algorithm. There are two attributes to OutputParm, outputName and parameterName. ParameterName is a parameter reference name for a parameter that will be updated by this algorithm. OutputName is an optional "friendly" name for the output parameter.

**AlgorithmText**

This optional element may be used to enter Pseudo or actual code for the algorithm. The language for the algorithm is specified with the language attribute.

**ExternalAlgorithm**

This is the external algorithm. Multiple entries are provided so that the same database may be used for multiple implementations.

### namespace
http://www.omg.org/space/tcspec

### children
LongDescription Trigger Inputs OutputParameterRef AlgorithmText ExternalAlgorithm

### used by
tc:CommandDefinitionType/CommandProcessor tc:CommandDefinitionType/CommandValidator tc:ComplexAlarmsType/CustomAlarmCheck SpaceSystemType/TelemetryAndCommanding/AlgorithmList/ CustomAlgorithm tc:CommandConstraintType/CustomAlgorithm tc:CommandVerifierType/CustomAlgorithm

### attributes
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>thread</td>
<td>boolean</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>triggerContainer</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>priority</td>
<td>integer</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
</tbody>
</table>

### documentation
The type definition used by the Custom Algorithm element.

Complete algorithm written in the algorithm language. Embedded new line characters are legal within this attribute.

All algorithms should contain a change log that is modified with each update.

A hint to the ground system to tell it to calculate the algorithm in a new processing thread.

### element
tc:CustomAlgorithmType/LongDescription
element **tc:CustomAlgorithmType/Trigger**

```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>name</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>triggerRate</td>
<td>nonNegativeInteger</td>
<td>optional</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
Names a parameter that will start the execution of the algorithm. There are two attributes to TriggerParm, triggerName and parameterName. parameterName is a parameter reference name for a parameter that when it changes, will cause this algorithm to be executed. triggerName is an optional "friendly" name for the trigger.
```

element **tc:CustomAlgorithmType/Inputs**

```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>inputName</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>constantName</td>
<td>nonNegativeInteger</td>
<td>optional</td>
<td>value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Names an input parameter to the algorithm. There are two attributes to InputParm, inputName and parameterName. parameterName is a parameter reference name for a parameter that will be used in this algorithm, inputName is an optional "friendly" name for the input parameter.

Names and provides a value for a constant input to the algorithm. There are two attributes to Constant, constantName and value, constantName is a variable name in the algorithm to be executed, value is the value of the constant to be used.
children ParameterRef Constant
element tc:CustomAlgorithmType/Inputs/ParameterRef

### Diagram

- **ParameterRef**
  - **Name**
    - A parameter name reference.
  - **tc:ParameterRefType**
    - **Occurs**
      - Repetition group to include
    - **DependantOn**
      - Optional parameter set condition that must be true for this parameter to be valid

### Description

- **Name**
  - An input parameter to the algorithm.
  - There are two attributes to InputParm, inputName and parameterName. parameterName is a parameter reference name for a parameter that will be used in this algorithm. inputName is an optional "friendly" name for the input parameter.

### Attributes

- **parameterName**
  - tc:NameReferenceType
  - Required
- **spaceSystemName**
  - String
- **inputName**
  - String

### Documentation

Names an input parameter to the algorithm. There are two attributes to InputParm, inputName and parameterName. parameterName is a parameter reference name for a parameter that will be used in this algorithm. inputName is an optional "friendly" name for the input parameter.

---

element tc:CustomAlgorithmType/Inputs/Constant

### Diagram

- **Constant**

### Attributes

- **constantName**
  - String
  - Required
- **value**
  - String

### Documentation

Names and provides a value for a constant input to the algorithm. There are two attributes to Constant, constantName and value. constantName is a variable name in the algorithm to be executed. value is the value of the constant to be used.

---

element tc:CustomAlgorithmType/OutputParameterRef
element tc:CustomAlgorithmType/AlgorithmText

diagram

This optional element may be used to enter Pseudo or actual code for the algorithm. The language for the algorithm is specified with the language attribute

extension of string
type attributes Name     Type     Use        Default     Fixed     Annotation

documentation This optional element may be used to enter Pseudo or actual code for the algorithm. The language for the algorithm is specified with the language attribute

element tc:CustomAlgorithmType/ExternalAlgorithm

diagram

This is the external algorithm. Multiple entries are provided so that the same database may be used for multiple implementations

extension of tc:ParameterRefType

Occurrences Dependent On

attributes Name     Type     Use        Default     Fixed     Annotation

occurrences     tc:NameReferenceType     required

parameterName

spaceSystemName     string

outputName     string

annotation
documentation Names an output parameter to the algorithm. There are two attributes to OutputParm, outputName and parameterName. parameterName is a parameter reference name for a parameter that will be updated by this algorithm. outputName is an optional "friendly" name for the output parameter.
<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>implementationName</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>algorithmLocation</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**documentation** This is the external algorithm. Multiple entries are provided so that the same database may be used for multiple implementations.
complexType `tc:MathAlgorithmType`

- A simple mathematical operation

namespace `http://www.omg.org/space/tcspec`

type extension of `tc:MathOperationType`

children

- ParameterRef
- Value
- Operator
- ParameterRef
- Value
- OutputParameterRef
- Trigger

used by

- `SpaceSystemType/TelemetryAndCommanding/AlgorithmList/MathAlgorithm`

attributes

- Name: `tc:NameReferenceType`
- required
- Default
- Fixed
- Annotation

annotation
documentation  A simple mathematical operation

element `tc:MathAlgorithmType/OutputParameterRef`

- type `tc:ParameterRefType`
- Occurs
- DependantOn

children

- Name: `tc:NameReferenceType`
- required
- spaceSystemName: `string`

attributes

- parameterName
- Use
- Default
- Fixed
- Annotation
element `tc:MathAlgorithmType/Trigger`

complexType `tc:ReedSolomonType`

```xml
<complexType name="tc:ReedSolomonType">
    <complexContent>
        <restriction base="tc:MathAlgorithmType">
            <choice>
                <element name="InputParameterRef" type="tc:NameReferenceType"/>
                <element name="OutputParameterRef" type="tc:NameReferenceType"/>
                <element name="GeneratorPolynomial" type="tc:NameReferenceType"/>
            </choice>
        </restriction>
    </complexContent>
</complexType>
```

```xml
<element name="tc:ReedSolomonType/InputParameterRef"/>
```

```xml
<element name="tc:ReedSolomonType/OutputParameterRef"/>
```

```xml
<element name="tc:ReedSolomonType/GeneratorPolynomial"/>
```
element `tc:ReedSolomonType/OutputParameterRef`

diagram

```
InputParameterRef

```

**type attributes**
- **Occurrences**: `0..∞`
- **DependantOn**: Optional parameter set condition that must be true for this parameter to be valid

**attributes**
- **parameterName**: `tc:NameReferenceType`
- **spaceSystemName**: `string`

```
OutputParameterRef

```

**type attributes**
- **Occurrences**: `0..∞`
- **DependantOn**: Optional parameter set condition that must be true for this parameter to be valid

**attributes**
- **parameterName**: `tc:NameReferenceType`
- **spaceSystemName**: `string`

```
GeneratorPolynomial

```

**type children**
- **Term**: A term in a polynomial expression.

**complexType** `tc:TriggerType`
A trigger is used to initiate the processing of some algorithm. A trigger may be based on an update of a Parameter or on a time basis. Triggers may also have a rate that limits their firing to a 1/rate basis.

http://www.omg.org/space/tcspec

ParameterRef Time

metadata

elements  tc:CustomAlgorithmType/Trigger  tc:MathAlgorithmType/Trigger

attributes

Name                   Type            Use         Default          Fixed          Annotation
name                    string          optional     
triggerRate           nonNegativeInteger          optional       1

annotation
documentation

A trigger is used to initiate the processing of some algorithm. A trigger may be based on an update of a Parameter or on a time basis. Triggers may also have a rate that limits their firing to a 1/rate basis.

element tc:TriggerType/ParameterRef

diagram

ParametersRef Type

Name: a parameter that will start the execution of the algorithm. There are two attributes to TriggerParm, triggerName and parameterName. parameterName is a parameter reference name for a parameter that when it changes, will cause this algorithm to be executed. triggerName is an optional "friendly" name for the trigger.

extension of tc:ParameterRefType

Occurrences DependentOn

Names a parameter that will start the execution of the algorithm. There are two attributes to TriggerParm, triggerName and parameterName. parameterName is a parameter reference name for a parameter that when it changes, will cause this algorithm to be executed. triggerName is an optional "friendly" name for the trigger.
element **tc:TriggerType/Time**

```
<Time type="tc:RelativeTimeType"/>
```

element **tc:DwellSet**

```
<tc:DwellSet/>
```

complexType **tc:ComplexAlarmsType**

```
<tc:ComplexAlarmsType/>
```

namespace http://www.omg.org/space/tcspec

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>onboardID</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>address</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>startBit</td>
<td>nonNegativeInteger</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>length</td>
<td>nonNegativeInteger</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ParameterName</td>
<td>tc:NameReferenceType</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>alternateParameterName</td>
<td>tc:NameReferenceType</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>annotation</td>
<td>documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**documentation**

Dwelling parameters

```
StaticRanges  RateOfChangeRanges  States  CustomAlarmCheck
```

Alarms are used to notify the operator when a parameter value is outside of acceptable ranges or states. Dynamic or rate-of-change Limit Ranges could be considered.

**StaticRanges**

A StaticRange has an optional UnitRangeWhen condition and one or more AlarmRanges.

**RateOfChangeRanges**

Like a Static Range, but it is based on the rate of change for the Parameter.

**States**

Contains multiple (un)acceptable values for multi-state parameters.

**CustomAlarmCheck**

An external algorithm that can be arbitrarily complex, e.g, an alarm that occurs when the parameter’s FFT shows a resonance at 20Hz.

```
<tc:ParameterType/AlarmConditions/>
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>minViolations</td>
<td>integer</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>limitsApplyToCalibrated</td>
<td>boolean</td>
<td></td>
<td></td>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>
```

**documentation**

Doesn’t get reported until it has been out of range for this value times true
Alarms are used to notify the operator when a parameter value is outside of acceptable ranges or states. Dynamic or rate-of-change Limit Ranges could be considered.
element tc:ComplexAlarmsType/StaticRanges

diagram

tc:RangeListType

StaticRanges

A StaticRange has an optional UseRangeWhen condition and one or more AlarmRanges.

UseRangeWhen

AlarmRange

1..∞

type tc:RangeListType

children

UseRangeWhen

AlarmRange

attributes

Name

Type

Use

Default

Fixed

Annotation

limitSetPriority:

nonNegativeInteger:

optional

annotation
documentation

A StaticRange has an optional UseRangeWhen condition and one or more AlarmRanges.

rateOfChangeRanges:

diagram

RateOfChangeRanges

Like a Static Range, but it is based on the rate of change for the Parameter.

UseRangeWhen

AlarmRange

1..∞

extension of tc:RangeListType

children

UseRangeWhen

AlarmRange

attributes

Name

Type

Use

Default

Fixed

Annotation

limitSetPriority:

nonNegativeInteger:

optional

perTimeBase:

string:

optional

second:

annotation
documentation

Rate of change alarms must be reference to some relative time base e.g. 3.4 volts per second

States:

diagram

States

Contains multiple (un)acceptable values for multi-state parameters.

UseRangeWhen

State

1..∞

(Un)acceptable values for multi-state parameters.
documentation  Contains multiple (un)acceptable values for multi-state parameters.
element tc:ComplexAlarmsType/States/UseRangeWhen

UseRangeWhen

EqualityCheck

CustomCheck

Can be used to implement conditional limits

documentation

Can be used to implement conditional limits


element tc:ComplexAlarmsType/States/UseRangeWhen/EqualityCheck

diagram

EqualityCheck

ComparisonOperator

ParameterRef

ArgumentRef

ParameterRef

Value

Type tc:ComparisonCheckType

ParameterRef

ComparisonOperator

ArgumentRef

ParameterRef

Value

Name calibratedComparison Type boolean Use false Default Annotation documentation

Should the comparison be based on the calibrated format or not, if true and a constant value is used this should be in calibrated format too.

element tc:ComplexAlarmsType/States/UseRangeWhen/CustomCheck

diagram

CustomCheck

element tc:ComplexAlarmsType/States/State
<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>statePriority</td>
<td>nonNegativeInteger</td>
<td>optional</td>
<td></td>
<td></td>
<td>(Un)acceptable values for multi-state parameters.</td>
</tr>
<tr>
<td>type</td>
<td>type</td>
<td>tc:LimitCheckType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state</td>
<td>state</td>
<td>nonNegativeInteger</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**diagram**

**State**
(Un)acceptable values for multi-state parameters.

**element** `tc:ComplexAlarmsType/CustomAlarmCheck`

**diagram**

```
<table>
<thead>
<tr>
<th>tc:CustomAlgorithmType</th>
</tr>
</thead>
<tbody>
<tr>
<td>LongDescription</td>
</tr>
<tr>
<td>Trigger</td>
</tr>
<tr>
<td>Inputs</td>
</tr>
<tr>
<td>OutputParameterRef</td>
</tr>
<tr>
<td>AlgorithmText</td>
</tr>
<tr>
<td>ExternalAlgorithm</td>
</tr>
</tbody>
</table>

**CustomAlarmCheck**
An external algorithm that can be arbitrarily complex. E.G., an alarm that occurs when the parameter's FFT shows a resonance at 20Hz.

**type** `tc:CustomAlgorithmType`

- **LongDescription**
- **Trigger**
- **Inputs**
- **OutputParameterRef**
- **AlgorithmText**
- **ExternalAlgorithm**

**children**

- LongDescription
- Trigger
- Inputs
- OutputParameterRef
- AlgorithmText
- ExternalAlgorithm
<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>name</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td></td>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td></td>
<td>thread</td>
<td>boolean</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td></td>
<td>triggerContainer</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>First telemetry container from which the output parameter should be calculated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>priority</td>
<td>integer</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Algorithm processing priority.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

annotation 

documentation An external algorithm that can be arbitrarily complex. E.G. an alarm that occurs when the parameter's FFT shows a resonance at 20Hz.
complexType `tc:OffsetParameterRefType`

namespace http://www.omg.org/space/tcspec
extension of `tc:ParameterRefType`

children
- `tc:OffsetParameterRefType`/`OffsetInBits`

used by
- `tc:SequenceType/ArgumentRef`
- `tc:OccursType/CountRef`
- `tc:SequenceType/DynamicParameterRef`
- `tc:DynamicContainerRefType/ParameterRef`
- `tc:SequenceType/ParameterRef`

attributes
- `parameterName` `tc:NameReferenceType` required
- `spaceSystemName` string
- `TimeOffset` integer optional 0

annotation
- documentation Holds an offset to a referred to parameter and optional comparisons to determine validity of this parameter

element `tc:OffsetParameterRefType/OffsetInBits`

`tc:BitOffsetType`

- `AbsoluteOffset` Absolute offset in bits. Default is 0.
- `RelativeOffset` Relative offset in bits from end of previous object, can be negative. Default is 0

annotation
- documentation Offset in bits from the start of the container scope, the first bit in the container has an offset of 0
complexType **tc:ParameterRefMatchListType**

```
complexType tc:ParameterRefMatchListType

  diagram

  tc:ParameterRefMatchListType -> ParameterMatch
  Holds a list of parameter matches, and whether all or one match required

  namespace http://www.omg.org/space/tcspec

  children

  ParameterMatch

  used by

  tc:ParameterRefType/DependantOn

  attributes

  Name                   Type                                  Use                      Default                       Fixed              Annotation
  AllMustBeTrue   boolean                                                                                                                   documentation

  documentation  Must all matched be true? If not present then all must be true

  element tc:ParameterRefMatchListType/ParameterMatch

  diagram

  tc:ComparisonCheckType

  ParameterRef

  ComparisonOperator

  ArgumentRef

  ParameterRef

  Value

  Condition that must be evaluated

  Takes the value of a command argument, is assumed to be of the same type as the comparison Parameter

  Parameter is assumed to be of the same type as the comparison Parameter

  Value is assumed to be of the same type as the comparison Parameter

  type tc:ComparisonCheckType

  ParameterRef ComparisonOperator ArgumentRef ParameterRef Value

  attributes

  Name                   Type                                  Use                      Default                       Fixed              Annotation
documentedCompared  boolean                                                false                                                 documentation

  documentation  Should the comparison be based on the calibrated format or not, if true and a constant value is used this should be in calibrated format too.

complexType **tc:ParameterRefType**
element \texttt{tc:ParameterRefType}/\texttt{Occurs}

documentation A reference to a Parameter that includes a validity check for the parameters
element `tc:ParameterRefType/DependantOn`

```xml
<Diagram>
  <DependantOn>
    <ParameterMatch />
  </DependantOn>
</Diagram>
```

**complexType `tc:ParameterRefMatchListType`**

**ParameterMatch**

**attributes**
- **Name**: `AllMustBeTrue`
- **Type**: `boolean`

**annotation**
- **documentation**: Must all matched be true? If not present then all must be true

**complexType `tc:ParameterType`**
td:BaseParameterType

- Any
- Float
- Integer
- Enumerated
  - The list of label/value pairs associated with a parameter of data type 'enumerated'
- Binary
- Boolean
- String
- AbsoluteTime
- RelativeTime
- Units
- Unitless
- LongDescription
  - An arbitrarily long description for the Parameter. May also include HTML markup.
- BusAttributes
- Alias
- SystemName
  - Optional. Normally used when the database is built in a flat, non-hierarchical format.
- ToString
  - Enforces instructions on how to represent the parameter as a string.
- Validity
- AlarmConditions
  - Alarm
- PhysicalAddress
  - Contains the address (channel information) needed to process the spacecraft telemetry streams.
- NonStandardData
  - 1

A type definition used by Parameter.
namespace http://www.omg.org/space/tcspec
extension of tc:BaseParameterType
children
Any Float Integer Enumerated Binary Boolean String AbsoluteTime RelativeTime Units Unitless LongDescription
BusAttributes Alias SystemName ToString Validity AlarmConditions Alarms PhysicalAddress NonStandardData
used by
element SpaceSystemType/TelemetryAndCommanding/ParameterList/Parameter
complexType tc:VariableParameterType
attributes
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>tc:NameType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dataSource</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>readOnly</td>
<td>boolean</td>
<td>optional</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>validRangeAppliesToCalibrated</td>
<td>boolean</td>
<td>optional</td>
<td>true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>onboardID</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
annotation documentation A type definition used by Parameter
element `tc:ParameterType/Alias`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>nameSet</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>alias</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

diagram

`tc:AliasType`

`Optional. Normally used when the database is built in a flat, non-hierarchical format`

element `tc:ParameterType/SystemName`

`tc:SystemNameType`

`Optional. Normally used when the database is built in a flat, non-hierarchical format`

element `tc:ParameterType/ToString`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueEnumeration</td>
<td>A number or range assigned to a string.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RangeEnumeration</td>
<td>A string value associated with a numerical range.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NumberFormat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

diagram

`tc:ToStringType`

`Includes instructions on how to represent the parameter as a string.`

element `tc:ParameterType/Validity`
element tc:ParameterType/AlarmConditions

Should the comparison be based on the calibrated format or not, if true and a constant value is used this should be in calibrated format too.
**Diagram 1: tc:ComplexAlarmsType**

- **StaticRanges**: A StaticRange has an optional UseRangeWhen condition and one or more AlarmRanges.
- **RateOfChangeRanges**: Like a Static Range, but it is based on the rate of change for the Parameter.
- **States**: Contains multiple (un)acceptable values for multi-state parameters.
- **CustomAlarmCheck**: An external algorithm that can be arbitrarily complex. E.g., an alarm that occurs when the parameter’s FFT shows a resonance at 20Hz.

**Attributes**
- **Name**
- **Type**
- **Use**
- **Default**
- **Fixed**
- **Annotation**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>minViolations</td>
<td>integer</td>
<td>required</td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>limitsApplyToCalibrated</td>
<td>boolean</td>
<td></td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram 2: tc:SimpleAlarmsType**

- **CriticalRange**
- **WarningRange**

**Element**

- **tc:ParameterType/Alarms**

- **tc:SimpleAlarmsType**

- **tc:ParameterType/PhysicalAddress**
element `tc:ParameterType/NonStandardData`

complexType `tc:PhysicalAddressType`

namespace `http://www.omg.org/space/tcspec`

used by
- `tc:ParameterType/PhysicalAddress`
- `tc:PhysicalAddressType/SubAddress`

attributes
- `sourceName` : string
- `sourceAddress` : string

annotation
documentation: When it's important to know the physical address(s) on the spacecraft that this parameter may be collected from, use this.

element `tc:PhysicalAddressType/SubAddress`
complexType `tc:RangeListType`

```
complexType tc:RangeListType

subType tc:PhysicalAddressType

SubAddress

attributes
  sourceName string
  sourceAddress string

UseRangeWhen

uses tc:ComplexAlarmsType/RateOfChangeRanges tc:ComplexAlarmsType/StaticRanges

attributes
  limitSetPriority nonNegativeInteger optional

annotation documentation Contains multible ranges
```

element `tc:RangeListType/UseRangeWhen`

```
UseRangeWhen

EqualityCheck

CustomCheck

children

annotation EqualityCheck CustomCheck

documentation Can be used to implement conditional limits
```

element `tc:RangeListType/UseRangeWhen/EqualityCheck`
element `tc:RangeListType/UseRangeWhen/CustomCheck`

```
CustomCheck
```

element `tc:RangeListType/AlarmRange`

```
AlarmRange
```

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>calibratedComparison</td>
<td>boolean</td>
<td>false</td>
<td>documentation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Should the comparison be based on the calibrated format or not, if true and a constant value is used this should be in calibrated format too.
<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>minInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>minExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>maxInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>maxExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>calibrated</td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>severity</td>
<td>nonNegativeInteger</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>type</td>
<td>tc:LimitCheckType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rangeName</td>
<td>string</td>
<td></td>
<td>optional</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
complexType tc:SimpleAlarmsType

diagram

A simple warning, critical alarm type

namespace http://www.omg.org/space/tcspec

children

CriticalRange  WarningRange

used by

element tc:ParameterType/Alarms

annotation
documentation A simple warning, critical alarm type

element tc:SimpleAlarmsType/CriticalRange

diagram

type tc:RangeType

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>minInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calibrated</td>
<td>boolean</td>
<td></td>
<td></td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

element tc:SimpleAlarmsType/WarningRange

diagram

type tc:RangeType

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>minInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calibrated</td>
<td>boolean</td>
<td></td>
<td></td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

complexType tc:StopType

diagram

Specifies either a width in
bits or a termination for
variable width parameters

namespace http://www.omg.org/space/tcspec

used by

element tc:VariableParameterType/stop

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>representation</td>
<td>hexBinary</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>widthInBits</td>
<td>positiveInteger</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

annotation
documentation Specifies either a width in bits or a termination for variable width parameters
complexType tc:VariableParameterType
namespace http://www.omg.org/space/tcspec
extension of tc:ParameterType
Any Float Integer Enumerated Binary Boolean String AbsoluteTime RelativeTime Units Unitless LongDescription
BusAttributes Alias SystemName ToString Validity AlarmConditions Alarms PhysicalAddress NonStandardData

WidthRef stop

attributes
Name                   Type                                  Use                      Default                       Fixed              Annotation

name                   tc:NameType                      required
shortDescription   tc:ShortDescriptionType      optional
dataSource          string                                   optional             false
readOnly             boolean                               optional             true
validRangeAppliesToCalibrated   boolean     optional             true
onboardID          string                                    optional

annotation
documentation A special parameter that may have a variable width
element `tc:VariableParameterType/WidthRef`

diagram

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WidthRef</td>
<td>tc:ParameterRefType</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Occurs</td>
<td>DependantOn</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0,∞</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Repeating group to include</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Optional parameter set condition that must be true for this parameter to be valid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**type**

- `tc:ParameterRefType`

**children**

- `tc:ParameterRefType`
- `Occurs` DependantOn

**attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterName</td>
<td>tc:NameReferenceType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spaceSystemName</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

element `tc:VariableParameterType/stop`

diagram

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>stop</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**type**

- `tc:StopType`

**attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>representation</td>
<td>hexBinary</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>widthInBits</td>
<td>positiveInteger</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**complexType** `tc:AbsoluteTimeType`

diagram

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>tc:AbsoluteTimeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contains an absolute (to a known epoch) time</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**namespace**

- http://www.omg.org/space/tcspec

**extension of** `dateTime`

**type**

- documentation  Contains an absolute (to a known epoch) time

**complexType** `tc:AliasType`
Used to save an alias (alternate) name or ID that may be used to identify the parameter by the factory or in an alternative ground system. Some ground system processing equipment has some severe naming restrictions on parameters (e.g., names must less than 12 characters, single case or integral id's only).
namespace http://www.omg.org/space/tcspec

element tc:BaseDataType

annotation
documentation: Used to describe the base data types used by the ground system. The ValidRange is a range for the parameter's value that is valid. It is used to display graphs and/or gauges that have pre-defined boundaries.

children
Any, Float, Integer, Enumerated, Binary, Boolean, String, AbsoluteTime, RelativeTime, Units, Unitless

used by
tc:BaseParameterType

complexType

Any, Float, Integer, Enumerated, Binary, Boolean, String, AbsoluteTime, RelativeTime, Units, Unitless

sources

SourceParameterRef

annotation
documentation: A place holder parameter that is populated at runtime by using the value currently contained in the SourceParameter as an onboard Id.
element tc:BaseDataType/Any/SourceParameterRef

diagram

\[\text{SourceParameterRef}\]

type tc:NameReferenceType

facets

pattern [a-zA-Z0-9_-/*]*

element tc:BaseDataType/Float

diagram

\[
\begin{array}{c}
\text{Float} \\
\text{defaultValue} \\
\text{validRange}
\end{array}
\]

children DefaultValue ValidRange

element tc:BaseDataType/Float/DefaultValue

diagram

\[
\text{DefaultValue}
\]

type tc:DefaultValueType

attributes

Name                   Type                                  Use                      Default                       Fixed              Annotation

calibrated             boolean                                                         false

element tc:BaseDataType/Float/ValidRange

diagram

\[
\text{ValidRange}
\]

type tc:RangeType

attributes

Name                   Type                                  Use                      Default                       Fixed              Annotation

minInclusive         string

minExclusive        string

maxInclusive        string

maxExclusive       string

calibrated             boolean                                                         false

element tc:BaseDataType/Integer

diagram

\[
\begin{array}{c}
\text{Integer} \\
\text{defaultValue} \\
\text{validRange}
\end{array}
\]

children DefaultValue ValidRange

element tc:BaseDataType/Integer/DefaultValue
### element `tc:BaseDataType/Integer/ValidRange`

The list of label/value pairs associated with a parameter of datatype 'integer'.

### element `tc:BaseDataType/Enumerated`

The list of label/value pairs associated with a parameter of datatype 'enumerated'.

### element `tc:BaseDataType/Enumerated/DefaultValue`

The list of label/value pairs associated with a parameter of datatype 'enumerated'.

### element `tc:BaseDataType/Enumerated/ValidRange`

The list of label/value pairs associated with a parameter of datatype 'enumerated'.
<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>minInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>minExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>maxInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>maxExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>calibrated</td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
element tc:BaseDataType/Binary

diagram

children DefaultValue ValidRange

attributes Name Type Use Default Fixed Annotation
lengthInBits nonNegativeInteger

element tc:BaseDataType/Binary/DefaultValue

diagram

type tc:DefaultValueType

attributes Name Type Use Default Fixed Annotation
calibrated boolean false

element tc:BaseDataType/Binary/ValidRange

diagram

type tc:RangeType

attributes Name Type Use Default Fixed Annotation
minInclusive string
minExclusive string
maxInclusive string
maxExclusive string
calibrated boolean false

element tc:BaseDataType/Boolean

diagram

children DefaultValue ValidRange

element tc:BaseDataType/Boolean/DefaultValue

diagram

type tc:DefaultValueType

attributes Name Type Use Default Fixed Annotation
calibrated boolean false
element `tc:BaseDataType/Boolean/ValidRange`

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>minInclusive</code></td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>minExclusive</code></td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>maxInclusive</code></td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>maxExclusive</code></td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>calibrated</code></td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```


element `tc:BaseDataType/String`

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>maxLength</code></td>
<td>nonNegativeInteger</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```


element `tc:BaseDataType/String/DefaultValue`

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>calibrated</code></td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```


element `tc:BaseDataType/String/Enumeration`

```
<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>string</td>
</tr>
</tbody>
</table>
```
Use the [ISO 8601] extended format CCYY-MM-DDTh:mm:ss where "CC" represents the century, "YY" the year, "MM" the month and "DD" the day, preceded by an optional leading "+" sign to indicate a negative number. If the sign is omitted, "+" is assumed. The letter "T" is the date/time separator and "hh", "mm", "ss" represent hour, minute and second respectively. Additional digits can be used to increase the precision of fractional seconds if desired i.e the format ss.ss... with any number of digits after the decimal point is supported.


**element tc:BaseDataType/AbsoluteTime/ValidRange**

```plaintext
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>minInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calibrated</td>
<td>boolean</td>
<td></td>
<td></td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>
```

**element tc:BaseDataType/RelativeTime**

**Default Value**

Duration’s are expressed as

\[ PnYnMnDnTnHnMnSnS, \]

where \( nY \) represents the number of years, \( nM \) the number of months, \( nD \) the number of days, \( 'T' \) is the date/time separator, \( nH \) the number of hours, \( nM \) the number of minutes and \( nS \) the number of seconds. The number of seconds can include decimal digits to arbitrary precision. For example, to indicate a duration of 1 year, 2 months, 3 days, 10 hours, and 30 minutes, one would write \( \text{P1Y2M3DT10H30M} \). One could also indicate a duration of minus 120 days as \( \text{-P120D} \).
Duration's are expressed as PnYnMnDTnHnMnSnS, where nY represents the number of years, nM the number of months, nD the number of days, 'T' is the date/time separator, nH the number of hours, nM the number of minutes and nS the number of seconds. The number of seconds can include decimal digits to arbitrary precision. For example, to indicate a duration of 1 year, 2 months, 3 days, 10 hours, and 30 minutes, one would write: P1Y2M3DT10H30M. One could also indicate a duration of minus 120 days as: -P120D.
element tc:BaseDataType/Unitless

diagram Unit

type tc:UnitType

attributes

Name               Type     Use   Default   Fixed   Annotation
power              decimal   optional 1
description        string

complexType tc:BaseParameterType
A type definition used as the base type for a CommandDefinition and for BusCommandDefinition.
element tc:BaseParameterType/BusAttributes

complexType tc:BusAttributesType

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitLength</td>
<td>unsignedInt</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bitOrder</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td>MostSignificantBitFirst</td>
</tr>
<tr>
<td>busType</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

complexType tc:BitOffsetType

annotation documentation Denotes a bit field in a message that can be used to identify a particular message.

namespace http://www.omg.org/space/tcspec

children AbsoluteOffset RelativeOffset

used by tc:OffsetParameterRefType/OffsetInBits tc:OccursType/OffsetInBits tc:MessageKeyType/OffsetInBits tc:ContainerRefType/OffsetInBits tc:DynamicContainerRefType/OffsetInBits tc:SequenceType/BinaryConstant/OffsetInBits

annotation documentation Denotes a bit field in a message that can be used to identify a particular message.

documentation An arbitrarily long description for the Parameter. May also include HTML markup.
element tc:BitOffsetType/AbsoluteOffset

diagram AbsoluteOffset

Absolute offset in bits. Default is 0.

type nonNegativeInteger

annotation documentation Absolute offset in bits. Default is 0.


element tc:BitOffsetType/RelativeOffset

diagram RelativeOffset

Relative offset in bits from end of previous object, can be negative. Default is 0.

type integer

annotation documentation Relative offset in bits from end of previous object, can be negative. Default is 0

complexType tc:BusAttributesType

diagram

Bus attributes describe how a particular piece of data is sent or received from some non-native, off-platform device. (e.g. a spacecraft)

namespace http://www.omg.org/space/tcspec

children ErrorDetectCorrect DefaultCalibrator ByteOrder ContextCalibrator

elements tc:BaseParameterType/BusAttributes SpaceSystemType/TelemetryAndCommanding/DefaultBusAttributes

attributes Name Type Use Default Fixed Annotation

bitLength unsignedInt required

bitOrder string MostSignificantBitFirst

busType string required

annotation documentation Bus attributes describe how a particular piece of data is sent or received from some non-native, off-platform device. (e.g. a spacecraft)

element tc:BusAttributesType/ErrorDetectCorrect
**tc:ErrorDetectCorrectType**

- **Parity**
  - Bit position starts with 'zero'.
- **CRC**
  - Cyclic Redundancy Check definition. Legal values for coefficients are 0 or 1; exponents must be integer values.

**tc:CalibratorType**

- **SplineCalibrator**
  - A calibration type where a segmented line in a raw vs calibrated plane is described using a set of points. Raw values are converted to calibrated values by finding a position on the line corresponding to the raw value. The algorithm triggers on the input parameter.
- **PolynomialCalibrator**
  - A calibration type where a curve in a raw vs calibrated plane is described using a set of polynomial coefficients. Raw values are converted to calibrated values by finding a position on the curve corresponding to the raw value. The first coefficient belongs with the \(X^0\) term, the next coefficient belongs to the \(X^1\) term and so on. The algorithm triggers on the condition parameter.
- **ToStringCalibrator**
  - A calibration type where the bus value is converted to a string value.

**element tc:BusAttributesType/DefaultCalibrator**

**attributes**

- **Name**
  - type: string
  - use: optional
  - default: not specified
  - fixed: not specified
  - annotation: not specified

- **shortDescription**
  - type: tc:ShortDescriptionType
  - use: optional
  - default: not specified
  - fixed: not specified
  - annotation: not specified
element `tc:BusAttributesType/ByteOrder`

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ByteOrder</td>
<td>Byte</td>
</tr>
</tbody>
</table>
```

Used to describe an arbitrary byte order in multibyte parameters. `order` is the order the byte is transmitted.

**children** Byte

**annotation documentation** Used to describe an arbitrary byte order in multibyte parameters. `order` is the order the byte is transmitted.

---

**element `tc:BusAttributesType/ByteOrder/Byte`**

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td></td>
</tr>
</tbody>
</table>
```

**attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>appearanceInStream</td>
<td>nonNegativeInteger</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>byteSignificance</td>
<td>nonNegativeInteger</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**element `tc:BusAttributesType/ContextCalibrator`**

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ContextCalibrator</td>
<td>Context</td>
</tr>
</tbody>
</table>
```

Use when different calibrations must be used on the Parameter in different contexts. Use the first one that tests true.

**children** Context Calibrator

**annotation documentation** Use when different calibrations must be used on the Parameter in different contexts. Use the first one that tests true.

---

**element `tc:BusAttributesType/ContextCalibrator/Context`**
element **tc:BusAttributesType/ContextCalibrator/Calibrator**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>calibratedComparison</td>
<td>boolean</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>documentation</td>
</tr>
</tbody>
</table>

Should the comparison be based on the calibrated format or not, if true and a constant value is used this should be in calibrated format too.
complexType `tc:ComparisonCheckType`

**Diagram**

- `tc:CalibratorType`
  - **SplineCalibrator**
    - A calibration type where a segmented line in a raw vs calibrated plane is described using a set of points. Raw values are converted to calibrated values by finding a position on the line corresponding to the raw value. The algorithm triggers on the input parameter.
  - **PolynomialCalibrator**
    - A calibration type where a curved line in a raw vs calibrated plane is described using a set of polynomial coefficients. Raw values are converted to calibrated values by finding a position on the curve corresponding to the raw value. The first coefficient belongs with the $X^0$ term, the next coefficient belongs to the $X^1$ term and so on. The algorithm triggers on the condition parameter.
  - **ToStringCalibrator**
    - A calibration type where the bus value is converted to a string value.

**Table**

<table>
<thead>
<tr>
<th>attribute</th>
<th>type</th>
<th>use</th>
<th>default</th>
<th>fixed</th>
<th>annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>optional</td>
<td>default</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td><code>tc:ShortDescriptionType</code></td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Diagram**

```
ParameterRef
  |
  `-------- ComparisonOperator

ParameterRef
  |
  `-------- ArgumentRef

ParameterRef
  |
  `-------- Value
```

**Namespace**

`http://www.omg.org/space/tcspec`

**Children**

- `ParameterRef`
- `ComparisonOperator`
- `ArgumentRef`
- `ParameterRef`
- `Value`

**Used by**

- `tc:CommandConstraintType/Comparison`
- `tc:CommandVerifierType/Comparison`
- `tc:BusAttributesType/ContextCalibrator/Context`
- `tc:ContainerRefType/DependantOn`
- `tc:DynamicContainerRefType/DependantOn`
- `tc:ComplexAlarmsType/States/UseRangeWhen/EqualityCheck`
- `tc:ParameterRefMatchListType/ParameterMatch`
- `tc:ParameterType/Validity`

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>calibratedComparison</td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td></td>
<td>Should the comparison be based on</td>
<td></td>
<td></td>
<td></td>
<td>the calibrated format or not, if</td>
</tr>
<tr>
<td></td>
<td>the calibrated format or not, if true</td>
<td></td>
<td></td>
<td></td>
<td>true and a constant value is used</td>
</tr>
<tr>
<td></td>
<td>and a constant value is used this</td>
<td></td>
<td></td>
<td></td>
<td>should be in calibrated format</td>
</tr>
<tr>
<td></td>
<td>should be in calibrated format too.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Annotation**

- documentation: A simple boolean expression

**Element**

`tc:ComparisonCheckType/ParameterRef`

**Diagram**

```
ParameterRef
```

**Type**

`tc:SimpleParameterRefType`

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterName</td>
<td>tc:NameReferenceType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spaceSystemName</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Element**

`tc:ComparisonCheckType/ComparisonOperator`

**Diagram**

```
ComparisonOperator
```

**Type**

`tc:ComparisonOperatorsType`
facets
  enumeration ==
  enumeration !=
  enumeration &lt;
  enumeration &lt;=
  enumeration >
  enumeration &gt;=

element tc:ComparisonCheckType/ArgumentRef

diagram

<table>
<thead>
<tr>
<th>ArgumentRef</th>
</tr>
</thead>
<tbody>
<tr>
<td>Takes the value of a command argument, is assumed to be of the same type as the comparison Parameter</td>
</tr>
</tbody>
</table>

type tc:SimpleParameterRefType

| parameterName | tc:NameReferenceType      | required |
| spaceSystemName | string                   |          |

annotation
documentation Takes the value of a command argument, is assumed to be of the same type as the comparison Parameter

element tc:ComparisonCheckType/ParameterRef

<table>
<thead>
<tr>
<th>ParameterRef</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter is assumed to be of the same type as the comparison Parameter</td>
</tr>
</tbody>
</table>

type tc:SimpleParameterRefType

| parameterName | tc:NameReferenceType      | required |
| spaceSystemName | string                   |          |

annotation
documentation Parameter is assumed to be of the same type as the comparison Parameter

element tc:ComparisonCheckType/Value

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value is assumed to be of the same type as the comparison Parameter</td>
</tr>
</tbody>
</table>

type string

annotation
documentation Value is assumed to be of the same type as the comparison Parameter

complexType tc:DefaultValueType

<table>
<thead>
<tr>
<th>tc:DefaultValueType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holds a default value and indicated whether it is calibrated or not</td>
</tr>
</tbody>
</table>

namespace http://www.omg.org/space/tcspec

used by tc:BaseDataType/Float/DefaultValue tc:BaseDataType/Integer/DefaultValue tc:BaseDataType/Enumerated/DefaultValue tc:BaseDataType/Boolean/DefaultValue tc:BaseDataType/Binary/DefaultValue tc:BaseDataType/String/DefaultValue tc:BaseDataType/AbsoluteTime/DefaultValue tc:BaseDataType/RelativeTime/DefaultValue

| calibrated | boolean | false |

annotation
documentation Holds a default value and indicated whether it is calibrated or not
complexType tc:ErrorDetectCorrectType

namespace http://www.omg.org/space/tcspec

children
  Parity
  CRC

used by
element tc:BusAttributesType/ErrorDetectCorrect
documentation A simple element that provides for simple, but common error checking and detection.

element tc:ErrorDetectCorrectType/Parity

documentation Bit position starts with 'zero'.

attributes
  Name | Type              | Use    | Default | Fixed | Annotation
  type  | string            | required |         |       |            
  bitsFromStart | nonNegativeInteger | required |         |       |            


element tc:ErrorDetectCorrectType/CRC

documentation Cyclic Redundancy Check definition. Legal values for coefficient's are 0 or 1. Exponents must be integer values.

attributes
  Name | Type              | Use    | Default | Fixed | Annotation
  PolynomialType | nonNegativeInteger |         |         |       |            

annotation documentation Cyclic Redundancy Check definition. Legal values for coefficient's are 0 or 1. Exponents must be integer values.
complexType tc:HeaderType

diagram

\[ \text{tc:HeaderType} \rightarrow \text{Author}^{0,\infty}
\rightarrow \text{Note}^{0,\infty}
\rightarrow \text{History}^{0,\infty} \]

Schema for a Header record. A header contains general information about the document or subdocument.

namespace http://www.omg.org/space/tcspec
children
Author Note History

used by:
SpaceSystemType/Header

attributes
Name                   Type               Use            Default       Fixed      Annotation
version                  string
date                      string
classification         string
documentation  Schema for a Header record. A header contains general information about the document or subdocument.

element tc:HeaderType/Author

diagram

\[ \text{Author} \]

type string

element tc:HeaderType/Note

diagram

\[ \text{Note} \]

type string

element tc:HeaderType/History

diagram

\[ \text{History} \]

type string
complexType `tc:IntegerRangeType`

```
complexType tc:IntegerRangeType

A range for integers
```

namespace http://www.omg.org/space/tcspec

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>long</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max</td>
<td>long</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

annotation documentation A range for integers

complexType `tc:MathOperationType`

```
complexType tc:MathOperationType

A simple math operation
```

namespace http://www.omg.org/space/tcspec

children `ParameterRef Value Operator ParameterRef Value`

used by `complexType tc:MathAlgorithmType`

annotation documentation A simple math operation

element `tc:MathOperationType/ParameterRef`

```
<ParameterRef>
```

type `tc:NameReferenceType`

definition pattern [a-zA-Z0-9_\-_/]*

element `tc:MathOperationType/Value`

```
<Value>
```

type `string`

definition Value is assumed to be of the same type as the comparison Parameter

annotation documentation Value is assumed to be of the same type as the comparison Parameter
element tc:MathOperationType/Operator

diagram

<table>
<thead>
<tr>
<th>Operator</th>
</tr>
</thead>
</table>

type tc:MathOperatorsType
facets

- enumeration +
- enumeration -
- enumeration mult
- enumeration div
- enumeration mod
- enumeration exp
- enumeration bitor
- enumeration bitand
- enumeration bitxor

element tc:MathOperationType/ParameterRef

diagram

<table>
<thead>
<tr>
<th>ParameterRef</th>
</tr>
</thead>
</table>

type tc:NameReferenceType
facets

- pattern [a-zA-Z0-9\-_/]*

annotation

string documentation Value is assumed to be of the same type as the comparison Parameter

complexType tc:NonStandardDataType

diagram

<table>
<thead>
<tr>
<th>tc:NonStandardDataType</th>
</tr>
</thead>
</table>

An any type that may be used in certain key locations within the database to hold mission unique data.

namespace http://www.omg.org/space/tcspec

type extension of anyType
	anyType

type used by

SpaceSystemType/TelemetryAndCommanding/NonStandardData SpaceSystemType/NonStandardData
tc:ParameterType/NonStandardData tc:CommandDefinitionType/NonStandardData
tc:PackagingDefinitionType/NonStandardData
tc:StreamType/NonStandardData
tc:NonStandardData

type attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>usage</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An any type that may be used in certain key locations within the database to hold mission unique data.
complexType `tc:OccursType`

```
Hold a structure that can be repeated X times, where X is held in the supplied parameter reference
```

namespace `http://www.omg.org/space/tcspec`

children

- `CountRef`
- `Count`
- `OffsetInBits`

used by

- `tc:ParameterRefType/Occurs`
- `tc:ContainerRefType/Occurs`
- `tc:DynamicContainerRefType/Occurs`
- `tc:SequenceType/Occurs`
- `tc:ChoiceType/Occurs`

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Name</code></td>
<td><code>NMTOKENS</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>parameterName</code></td>
<td><code>tc:NameReferenceType</code></td>
<td><code>required</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>spaceSystemName</code></td>
<td><code>string</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>TimeOffset</code></td>
<td><code>integer</code></td>
<td><code>optional</code></td>
<td><code>0</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

annotation

- `documentation` Hold a structure that can be repeated X times, where X is held in the supplied parameter reference

element `tc:OccursType/CountRef`

```
Parameter that contains the count of repeated structures, implies variable length structure
```

```
Optional parameter set condition that must be true for this parameter to be valid
```

```
Offset in bits from the start of the container scope, the first bit in the container has an offset of 0
```

```
Repeating group to include
```

```
Time delay in milliseconds
```

```
Parameter that contains the count of repeated structures, implies variable length structure
```
element `tc:OccursType/Count`  

**Diagram**  
Fixed value that contains the count of repeated structures, implies fixed length structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumOfOcc</td>
<td>positiveInteger</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>RelativeOffsetInBits</td>
<td>nonNegativeInteger</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>TimeOffset</td>
<td>integer</td>
<td>optional</td>
<td>0</td>
<td></td>
<td>documentation</td>
</tr>
</tbody>
</table>

**Documentation**  
- `NumOfOcc`: Number of occurrences of structure in container
- `RelativeOffsetInBits`: Number of bits between the start of two consecutive structures
- `TimeOffset`: Time delay in milliseconds between two consecutive occurrences

element `tc:OccursType/OffsetInBits`  

**Diagram**  
Offset in bits from the start of the container scope

**Type**  
`tc:BitOffsetType`  
- `AbsoluteOffset`: Absolute offset in bits. Default is 0.
- `RelativeOffset`: Relative offset in bits from end of previous object, can be negative. Default is 0.

**ComplexType**  
`tc:PolynomialType`  
A polynomial expression. For example: 3 + 2x

**Namespace**  
http://www.omg.org/space/tcspec

**Terms**  
- `tc:ReedSolomonType/GeneratorPolynomial`
- `tc:ConvolutionalStreamType/Polynomial`
- `tc:CalibratorType/PolynomialCalibrator`
- `tc:ErrorDetectCorrectType/CRC/PolynomialType`

**Annotation**  
A term in a polynomial expression.

**Example**  
A polynomial expression. For example: 3 + 2x
complexType tc:PropertyType

A term in a polynomial expression.

documentation A term in a polynomial expression.

complexType tc:RangeType

A range of numbers. "minInclusive", "minExclusive", "maxInclusive" and "maxExclusive" attributes are borrowed from the W3C schema language.
namespace http://www.omg.org/space/tcspec

elements
- tc:RangeListType/AlarmRange
- tc:SimpleAlarmsType/CriticalRange
- tc:ToStringType/RangeEnumeration/Range
- tc:BaseDataType/Float/ValidRange
- tc:BaseDataType/Integer/ValidRange
- tc:BaseDataType/Enumerated/ValidRange
- tc:BaseDataType/Binary/ValidRange
- tc:BaseDataType/Boolean/ValidRange
- tc:BaseDataType/AbsoluteTime/ValidRange
- tc:SimpleAlarmsType/WarningRange

attributes
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>minInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calibrated</td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

annotation
documentation A range of numbers. "minInclusive", "minExclusive", "maxInclusive" and "maxExclusive" attributes are borrowed from the W3C schema language.
complexType tc:RealRangeType

A range for real numbers. "minInclusive", "minExclusive", "maxInclusive" and "maxExclusive" attributes are borrowed from the W3C schema language.

complexType tc:SimpleParameterRefType

A simple reference to a Parameter

complexType tc:SplinePointType

A spline is a set of points from which a curve may be drawn to interpolate raw to calibrated values.
complexType `tc:SystemNameType`

```xml
<complexType name="SystemNameType">
  <documentation>Sub-system name.</documentation>
</complexType>
```

namespace `http://www.omg.org/space/tcspec`

type extension of `string`

diagram
- `SystemNameType`

used by `tc:ParameterType/SystemName` `tc:CommandDefinitionType/SystemName`

annotation
documentation Sub-system name.

complexType `tc:ToStringType`

```xml
<complexType name="ToStringType">
  <documentation>There are two ways numeric data can be changed to string data: using a C style format string, or using an enumerated list. Enumerated lists can be assigned to a single value or a value range.</documentation>
</complexType>
```

namespace `http://www.omg.org/space/tcspec`

type extension of `string`

diagram
- `ToStringType`

used by `tc:ParameterType/ToString` `tc:CalibratorType/ToStringCalibrator`

annotation
documentation There are two ways numeric data can be changed to string data: using a C style format string, or using an enumerated list. Enumerated lists can be assigned to a single value or a value range.

element `tc:ToStringType/ValueEnumeration`

```xml
<element name="ValueEnumeration">
  <documentation>A number or range assigned to a string.</documentation>
</element>
```

attributes
- Name
- Type: integer
- Use: required
- Default
- Fixed
- Annotation
documentation A number or range assigned to a string.

element `tc:ToStringType/RangeEnumeration`

```xml
<element name="RangeEnumeration">
  <documentation>A string value associated with a numerical range.</documentation>
</element>
```

children
- `Range`
<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>stringValue</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>annotation</td>
<td>documentation</td>
<td>A string value associated with a numerical range.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
element tc:ToStringType/RangeEnumeration/Range

diagram

**Range**

tc:RangeType

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>minInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxInclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxExclusive</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calibrated</td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

element tc:ToStringType/NumberFormat

diagram

**NumberFormat**

tc:RadixType

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>numberBase</td>
<td>tc:RadixType</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minimumFractionDigits</td>
<td>nonNegativeInteger</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maximumFractionDigits</td>
<td>nonNegativeInteger</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>minimumIntegerDigits</td>
<td>nonNegativeInteger</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maximumIntegerDigits</td>
<td>nonNegativeInteger</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>negativeSuffix</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>positiveSuffix</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>negativePrefix</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>positivePrefix</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>showThousandsGrouping</td>
<td>boolean</td>
<td>optional</td>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>notation</td>
<td>string</td>
<td>optional</td>
<td></td>
<td>normal</td>
<td></td>
</tr>
</tbody>
</table>

complexType tc:UnitType

diagram

**tc:UnitType**

namespace http://www.omg.org/space/tcspec

element tc:BaseDataType/Units/Unit

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>power</td>
<td>decimal</td>
<td>optional</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

annotation documentation Used to hold the unit(s) plus possibly the exponents for the units

simpleType tc:ComparisonOperatorsType

namespace http://www.omg.org/space/tcspec

element tc:ComparisonCheckType/ComparisonOperator

attribute tc:MessageKeyMatchType/@Operator
facets

- `enumeration ==`
- `enumeration !=`
- `enumeration <`
- `enumeration <=`
- `enumeration >`
- `enumeration >=`

annotation
documentation Operators to use when testing a boolean condition for a validity check
simpleType `tc:FormatType`

namespace `http://www.omg.org/space/tcspec`
restriction of `string`

enumeration `SignedInteger`
enumeration `UnsignedInteger`
enumeration `Real`

annotation
documentation `Add`

simpleType `tc:IdentificationNumberType`

namespace `http://www.omg.org/space/tcspec`
restriction of `integer`

minInclusive `1`

maxInclusive `9999`

annotation
documentation `Add`

simpleType `tc:LimitCheckType`

namespace `http://www.omg.org/space/tcspec`
restriction of `string`

enumeration `warning`
enumeration `critical`
enumeration `statusConsistency`
enumeration `eventGeneration`

annotation
documentation Identifies the type of limit set check

simpleType `tc:LongDescriptionType`

namespace `http://www.omg.org/space/tcspec`

string

used by `SpaceSystemType/LongDescription` `tc:CustomAlgorithmType/LongDescription` `tc:BaseParameterType/LongDescription` `tc:CommandDefinitionType/LongDescription` `tc:CommandDefinitionType/Field/LongDescription` `tc:MessageKeyType/LongDescription` `tc:MessageType/LongDescription` `tc:ContainerType/LongDescription` `tc:ServiceType/LongDescription` `tc:SequenceType/BinaryConstant/LongDescription` `tc:PCMStreamType/LongDescription`

annotation
documentation An unbounded description

simpleType `tc:MathOperatorsType`

namespace `http://www.omg.org/space/tcspec`
restriction of `string`

used by `tc:MathOperationType/Operator`
facets
  enumeration +
  enumeration -
  enumeration mult
  enumeration div
  enumeration mod
  enumeration exp
  enumeration bitor
  enumeration bitand
  enumeration bitxor

annotation
documentation Mathematical operators
simpleType `tc:NameReferenceType`

namespace http://www.omg.org/space/tcspec

type restriction of string

used by `tc:ServiceType/MessageList/NameRef`

`tc:SyncStrategyType/MessageMatch/NameRef`

`tc:MathOperationType/ParameterRef`

`tc:MathOperationType/ParameterRef`

`tc:BaseDataType/Any/SourceParameterRef`

attributes

`tc:DwellSet/@alternateParameterName`

`tc:ReedSolomonType/@inputParameterRef`

`tc:PropertyType/@name`

`tc:ReedSolomonType/@outputParameterRef`

`tc:DwellSet/@ParameterName`

`tc:SimpleParameterRefType/@parameterName`

`tc:MathAlgorithmType/@trigger`

facets

pattern `[a-zA-Z0-9-_\-\ ]*`

annotation
documentation Used when referencing a directory style "NameType".

simpleType `tc:NameType`

namespace http://www.omg.org/space/tcspec

restriction of string

simpleTypes `tc:ContainerNameType`

`tc:MessageKeyIdType`

`tc:MessageNameType`

`tc:ServiceNameType`

attributes

`tc:BaseParameterType/@name`

`tc:CommandDefinitionType/@name`

`tc:PCMStreamType/@name`

pattern `[a-zA-Z0-9 \-\ ]*`

documentation Used for "directory" style unique names. Only letters, digits, ‘_’, ‘ ’ and ‘-’ are allowed

simpleType `tc:RadixType`

namespace http://www.omg.org/space/tcspec

type restriction of string

attribute `tc:ToStringType/NumberFormat/@numberBase`

enumeration Decimal

enumeration Hexadecimal

enumeration Octal

enumeration Binary

annotation
documentation Specifies the number base

simpleType `tc:RelativeTimeType`

namespace http://www.omg.org/space/tcspec

duration

element `tc:TriggerType/Time`

documentation Used to describe a relative time. Normally used for time offsets. A Relative time is expressed as PnYn MnDTnH nMnS, where nY represents the number of years, nM the number of months, nD the number of days, 'T' is the date/time separator, nH the number of hours, nM the number of minutes and nS the number of seconds. The number of seconds can include decimal digits to arbitrary precision. For example, to indicate a duration of 1 year, 2 months, 3 days, 10 hours, and 30 minutes, one would write: P1Y2M3DT10H30M. One could also indicate a duration of minus 120 days as: -P120D.

simpleType `tc:ShortDescriptionType`
complexType `tc:ArgumentType`

- namespace: http://www.omg.org/space/tcspec
- restriction of: string
- used by: `SpaceSystemType/@shortDescription`
- `CustomAlgorithmType/@shortDescription`
- `CalibratorType/@shortDescription`
- `BaseParameterType/@shortDescription`
- `CommandDefinitionType/@shortDescription`
- `MessageKeyType/@ShortDescription`
- `MessageType/@shortDescription`
- `ContainerType/@shortDescription`
- `ServiceType/@shortDescription`
- `PCMStreamType/@shortDescription`

- facets: maxLength 32
- annotation: documentation  A bounded description

---

- `tc:BaseParameterType`
  - Any
  - Float
  - Integer
  - Enumerated
    - Binary
    - Boolean
    - String
    - AbsoluteTime
    - RelativeTime
    - Units
    - Unitless

- `tc:ArgumentType`:
  - A type definition used as the base type for a CommandDefinition and for BusCommandDefinition.
  - `LongDescription`:
    - An arbitrarily long description for the Parameter. May also include HTML markup.
  - `BusAttributes`
namespace http://www.omg.org/space/tcspec

type extension of tc:BaseParameterType

children
Any Float Integer Enumerated Binary Boolean String AbsoluteTime RelativeTime Units Unitless LongDescription BusAttributes Argument

used by elements tc:ArgumentType/Argument tc:CommandDefinitionType/Field/Argument SpaceSystemType/

 TelemetryAndCommanding/CommandDefinitionList/ArgumentDefinition

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>tc:NameType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

documentation A type definition used as the base type for a CommandDefinition and for BusCommandDefinition.
element tc:ArgumentType/Argument

complexType tc:CommandConstraintType
A command constraint is used to check that the command can be run in the current device operating mode or environment. Command constraints are the name of some external algorithm that does the command constraint check or a list of simple comparison checks.

Element `tc:CommandConstraintType/CustomAlgorithm`

- **LongDescription**
- **Trigger**
- **Inputs**
- **OutputParameterRef**
  - Names an output parameter to the algorithm. There are two attributes to `OutputParm`, `outputName` and `parameterName`. `parameterName` is a parameter reference name for a parameter that will be updated by the algorithm. `outputName` is an optional "friendly" name for the output parameter.
- **AlgorithmText**
  - This optional element may be used to enter Pseudo or actual code for the algorithm. The language for the algorithm is specified with the language attribute.
- **ExternalAlgorithm**
  - This is the external algorithm. Multiple entries are provided so that the same database may be used for multiple implementations.
### tc:CustomAlgorithmType

**LongDescription**, **Trigger**, **Inputs**, **OutputParameterRef**, **AlgorithmText**, **ExternalAlgorithm**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread</td>
<td>boolean</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>triggerContainer</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td>integer</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Documentation**: First telemetry container from which the output parameter should be calculated.
- **Annotation**: Algorithm processing priority.
element `tc:CommandConstraintType/Comparison`

```
    tc:ComparisonCheckType
      ParameterRef
      ComparisonOperator
      ArgumentRef
      ParameterRef
      Value

ParameterRef
ParameterRef
Value

Takes the value of a command argument, is assumed to be of the same type as the comparison Parameter.

Parameter is assumed to be of the same type as the comparison Parameter.

Value is assumed to be of the same type as the comparison Parameter.
```

<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>calibratedComparison</td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
</tbody>
</table>

Should the comparison be based on the calibrated format or not, if true and a constant value is used this should be in calibrated format too.

complexType `tc:CommandDefinitionType`
namespace http://www.omg.org/space/tcspec

children

LongDescription  Alias  SystemName  Interlock  Field  BinaryTransform  CommandProcessor  CommandValidator  CommandConstraint  CommandVerifier  ParametersToSet  NonStandardData

custom elements SpaceSystemType/TelemetryAndCommanding/CommandDefinitionList/CommandDefinition SpaceSystemType/

TelemetryAndCommanding/DefaultCommandDefinition

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>tc:NameType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>critical</td>
<td>boolean</td>
<td>optional</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>highPriority</td>
<td>boolean</td>
<td>optional</td>
<td>false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

documentation  A type definition used as the base type for a CommandDefinition
element `tc:CommandDefinitionType/LongDescription`

```
<diagram type='tc:LongDescriptionType'>
```

element `tc:CommandDefinitionType/Alias`

```
<diagram type='tc:AliasType'>
```

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>nameSet</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>alias</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

element `tc:CommandDefinitionType/SystemName`

```
<diagram type='tc:SystemNameType'>
```

element `tc:CommandDefinitionType/Interlock`

```
<diagram type='Interlock'>
```

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>tc:InterlockScopeType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stage</td>
<td>tc:StageType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

annotation
documentation

Specifies the interlock for this command, including the scope of the interlock. An interlock restricts commanding ability at the specified scope until this command has reached the specified stage.

element `tc:CommandDefinitionType/Field`
May also use a reference to a constant parameter
**element** `tc:CommandDefinitionType/Field/ArgumentRef`

**tc:ArgumentType**

- **Any**
  - A placeholder parameter that is populated at runtime by using the value currently contained in the SourceParameter as an onboard Id.

- **Float**
- **Integer**

- **Enumerated**
  - The list of label/value pairs associated with a parameter of datatype 'enumerated'.

- **Binary**
- **Boolean**
- **String**

- **AbsolutePath**
- **RelativeTime**

- **Units**
- **Unitless**

**BusAttributes**

- **LongDescription**
  - An arbitrarily long description for the Parameter. May also include HTML markup.
element `tc:CommandDefinitionType/Field/ParameterRef`

```
<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterName</td>
<td>tc:NameReferenceType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spaceSystemName</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
May also use a reference to a constant parameter
```

element `tc:CommandDefinitionType/Field/BinaryConstant`

```
<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>lengthInBits</td>
<td>nonNegativeInteger</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
May also use a reference to a constant parameter
```

element `tc:CommandDefinitionType/BinaryTransform`
The binary transform tells the system how to construct the command (transform it to binary format).

**Element**: `tc:CommandDefinitionType/BinaryTransform/FieldBitPosition`

**Diagram**

```
FieldBitPosition -- BitPositionFromStart
FieldBitPosition -- BitPositionFromPreviousArgument
```

**Attributes**

- `Name` (string)
- `Type` (nonNegativeInteger)
- `Use` (integer)
- `Default` (integer)
- `Fixed` (boolean)
- `Annotation` (string)

**Element**: `tc:CommandDefinitionType/BinaryTransform/FieldBitPosition/BitPositionFromStart`

**Type**: `nonNegativeInteger`

**Element**: `tc:CommandDefinitionType/BinaryTransform/FieldBitPosition/BitPositionFromPreviousArgument`

**Type**: `integer`

**Element**: `tc:CommandDefinitionType/BinaryTransform/ContainerRef`
element `tc:CommandDefinitionType/CommandProcessor`

**Diagram:**
- `tc:ContainerRefType` with attributes:
  - `nameRef`: `tc:ContainerNameType` (required)
  - `DependantOn`:
    - Parameter that the inclusion of the sub-container depends on.
  - `OffsetInBits`:
    - Offset, in bits, from the start of this container to start sub-container. If not present then sub-container start either at offset 0 if the first sub-container or relative to end of previous object.

**Type Children Attributes:**
- `nameRef`:
  - `Type`: `tc:ContainerNameType`
  - `Use`: required

**Diagram Legend:**
- Occurs: Repeating group to include
- DependantOn: Parameter that the inclusion of the sub-container depends on
- OffsetInBits: Offset, in bits, from the start of this container to start sub-container. If not present then sub-container start either at offset 0 if the first sub-container or relative to end of previous object.
A command processor is the name of some external algorithm that is used to format the command. Command processors may also perform additional formatting of the command - formatting that cannot be adequately described in the database.

This optional element may be used to enter Pseudo or actual code for the algorithm. The language for the algorithm is specified with the language attribute.

This is the external algorithm. Multiple entries are provided so that the same database may be used for multiple implementations.

**longDescription**: A command processor is the name of some external algorithm that is used to format the command. Command processors may also perform additional formatting of the command - formatting that cannot be adequately described in the database.
A command validator is used to validate that a particular user has the proper permissions to issue the command. Command Validators - like command processors - are the name of some external algorithm that does the command validation.

This optional element may be used to enter Pseudo or actual code for the algorithm. The language for the algorithm is specified with the language attribute.

This is the external algorithm. Multiple entries are provided so that the same database may be used for multiple implementations.
element `tc:CommandDefinitionType/CommandVerifier`

- **Diagram:**
  - `CommandVerifier` is connected to:
    - `CustomAlgorithm` with multiplicity `1..∞`
    - `Comparison` with multiplicity `1..∞`
    - `ContainerRef` with multiplicity `1..∞`
    - `Window`

- **Attributes:**
  - **Name**
  - **Type**
  - **Use**
  - **Default**
  - **Fixed**
  - **Annotation**

- **Children:**
  - `CustomAlgorithm`
  - `Comparison`
  - `ContainerRef`
  - `Window`

- **Annotation:**
  - Specifies how much of a chance to provide for the verification. The window may be specified in time or in number of samples of the comparison parameter to wait for a change.

element `tc:CommandDefinitionType/ParametersToSet`

- **Diagram:**
  - `ParametersToSet` is connected to:
    - `Value`
    - `ArgumentRef`

- **Annotation:**
  - Sets the values of parameters after the command has been verified

element `tc:CommandDefinitionType/ParametersToSet/ParameterRef`
element `tc:CommandDefinitionType/ParametersToSet/Value`

type `string`

element `tc:CommandDefinitionType/ParametersToSet/ArgumentRef`

type `string`

element `tc:CommandDefinitionType/NonStandardData`

complexType `tc:CommandVerifierType`
A command verifier is used to check that the command has been successfully executed. Command Verifiers are the name of some external algorithm that does the command verification.

namespace http://www.omg.org/space/tcspec

children CustomAlgorithm Comparison ContainerRef Window

attributes

- Name: verificationStage
- Type: tc:StageType
- Use: Completion
- Default: Fixed
- Annotation

annotation documentation A command verifier is used to check that the command has been successfully executed. Command Verifiers are the name of some external algorithm that does the command verification.

element tc:CommandVerifierType/CustomAlgorithm
**element** tc:CommandVerifierType/Comparison

**type** tc:CustomAlgorithmType

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread</td>
<td>boolean</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>triggerContainer</td>
<td>string</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td>integer</td>
<td>optional</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LongDescription**

Names an output parameter to the algorithm. There are two attributes to OutputParam, outputName and parameterName. parameterName is a parameter reference name for a parameter that will be updated by this algorithm. outputName is an optional "friendly" name for the output parameter.

**Trigger**

This optional element may be used to enter Pseudo or actual code for the algorithm. The language for the algorithm is specified with the language attribute.

**Inputs**

**OutputParameterRef**

0..∞

**AlgorithmText**

This is the external algorithm. Multiple entries are provided so that the same database may be used for multiple implementations.

**ExternalAlgorithm**

0..∞
element `tc:CommandVerifierType/ContainerRef`

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Attributes</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>calibratedComparison</code></td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td>documentation</td>
</tr>
</tbody>
</table>
```

Should the comparison be based on the calibrated format or not, if true and a constant value is used this should be in calibrated format too.

---

**Diagram**

```
tc:ComparisonCheckType
  ParameterRef
  ComparisonOperator
    ArgumentRef
      ParameterRef
      Value
```

- **ParameterRef**
  - Type: `tc:ParameterRef`
  - Description: Takes the value of a command argument, is assumed to be of the same type as the comparison Parameter

- **ComparisonOperator**
  - Description: Parameter is assumed to be of the same type as the comparison Parameter

- **Value**
  - Description: Value is assumed to be of the same type as the comparison Parameter

---

**Diagram**

```
tc:ContainerRefType
  ContainerRef
    Occurs
      0..\infty
    DependantOn
    OffsetInByte
      Offset, in bits, from the start of this container to start sub-container. If not present then sub-container start either at offset 0 if the first sub-container or relative to end of previous object
```

- **ContainerRef**
  - Type: `tc:ContainerRefType`
  - Description: Repeating group to include

- **DependantOn**
  - Description: Parameter that the inclusion of the sub-container depends on

- **OffsetInByte**
  - Description: Offset, in bits, from the start of this container to start sub-container. If not present then sub-container start either at offset 0 if the first sub-container or relative to end of previous object
<table>
<thead>
<tr>
<th>childen attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>nameRef</td>
<td></td>
<td>tc:ContainerNameType</td>
<td>required</td>
<td></td>
<td></td>
<td>Id of container to contain</td>
</tr>
</tbody>
</table>

**Occurrences DependantOn OffsetInBits**
**element** `tc:CommandVerifierType/Window`  

```
Diagram:
```

```
 startTime  interval  changeCount
```

**documentation**  Specifies how much of a chance to provide for the verification. The window may be specified in time or in number of samples of the comparison parameter to wait for a change.

**element** `tc:CommandVerifierType/Window/StartTime`  

```
Diagram:
```

**type** `nonNegativeInteger`

**element** `tc:CommandVerifierType/Window/Interval`  

```
Diagram:
```

**type** `positiveInteger`

**element** `tc:CommandVerifierType/Window/ChangeCount`  

```
Diagram:
```

**type** `positiveInteger`

**simpleType** `tc:InterlockScopeType`  

```
namespace       http://www.omg.org/space/tcspec
restriction     of    string
used by         attribute `tc:CommandDefinitionType/Interlock/@scope`
enumeration     SystemGlobal
enumeration     SystemLocal
enumeration     SubsystemGlobal
enumeration     SubsystemLocal
annotation     documentation Specifies the scope of the interlock
```
simpleType `tc:StageType`

namespace `http://www.omg.org/space/tcspec`
restriction of `string`
attributes `tc:CommandDefinitionType/Interlock/@stage` `tc:CommandVerifierType/@verificationStage`
enumeration Reception
enumeration Uplink
enumeration Transfer
enumeration Acceptance
enumeration Execution
enumeration Completion
annotation
documentation Specifies the stage of command uplink

element `tc:Packaging`

```
namespace `http://www.omg.org/space/tcspec`
type `tc:PackagingDefinitionType`
children `MessageList` `ContainerList` `NonStandardData`
used by element `SpaceSystemType/TelemetryAndCommanding`
annotation
documentation Root packaging element for this schema
```

complexType `tc:ChoiceType`

```
namespace `http://www.omg.org/space/tcspec`
type `tc:ChoiceType`/Occurs
children `Occurs` `ContainerRef`
used by element `tc:SequenceType/Choice`
annotation
documentation Used to one or more possible container choices.
```

element `tc:ChoiceType/Occurs`
**element tc:ChoiceType/ContainerRef**

- **Namespace**: http://www.omg.org/space/tcspec
- **Type**: tc:ContainerRefType
- **Children**: Count, OffsetsInBits
- **Attributes**:
  - Name
  - Type: NMTOKENS
  - Use
  - Default: Occurs

**Annotation**
- Documentation: Repeating group to include

---

**complexType tc:ContainerRefType**

- **Namespace**: http://www.omg.org/space/tcspec
- **Type**: tc:ContainerRefType
- **Children**: Occurs, DependantOn, OffsetsInBits
- **Attributes**:
  - NameRef: tc:ContainerNameType
  - Use: required

**Annotation**
- Documentation: Id of container to contain
element `tc:ContainerRefType/Occurs`

Diagram:

- **Occurs**: Repeating group to include
- **DependantOn**: Parameter that the inclusion of the sub-container depends on
- **OffsetInBits**: Offset, in bits, from the start of this container to start sub-container. If not present then sub-container start either at offset 0 if the first sub-container or relative to end of previous object.

Attributes:

- **nameRef**
  - Type: `tc:ContainerNameType`
  - Use: `required`

Annotation:

Holds an offset to a referred to container and an optional parameter to determine validity of this container.

```
namespace http://www.omg.org/space/tcspec
children
  type `tc:OccursType`
  children
    - **CountRef**
    - **Count**
      - Fixed value that contains the count of repeated structures, implies fixed length structure
    - **OffsetInBits**
      - Offset in bits from the start of the container scope

attributes

- **CountRef**
- **Count**
- **OffsetInBits**

annotation
documentation

Repeating group to include

```

```

```

```
element tc:ContainerRefType/OffsetInBits

documentation Parameter that the inclusion of the sub-container depends on

namespace http://www.omg.org/space/tcspec
type tc:ContainerRefType
children

annotation
documentation Offset, in bits, from the start of this container to start sub-container. If not present then sub-container start either at offset 0 if the first sub-container or relative to end of previous object
complexType `tc:ContainerType`

```
represented by `tc:ContainerType` diagram
```

namespace `http://www.omg.org/space/tcspec`

children

- LongDescription
- ImplementsMessage

LongDescription

- `tc:ContainerNameType` required
- `tc:ShortDescriptionType` optional

ImplementsMessage

- `tc:MessageNameType` required
- `boolean` `AllMustBeTrue`

annotation

```
Optional long description of container
```

```
Optional message this container represents
```

```
Optional body of this container. Most container will have a body however event packets do not.
```

```
Id of container
```

```
Optional short description of container
```

```
Documentation
```

```
Represents a packet/frame/sub-frame etc.
```

```
```

element `tc:ContainerType/LongDescription`

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```
element tc:ContainerType/Sequence

Optional body of this container. Most container will have a body however event packets do not.

complexType tc:DynamicContainerRefType

namespace http://www.omg.org/space/tcspec
tc:SequenceType

Occurs ContainerRef DynamicContainerRef ArgumentRef ParameterRef DynamicParameterRef BinaryConstant
Choice Sequence

documentation Optional body of this container. Most container will have a body however event packets do not.
element tc:DynamicContainerRefType/Occurs

Parameter that will hold container id at runtime

OffsetInBits
Offset, in bits, from the start of this container to start sub-container. If not present then sub-container start either at offset 0 if the first sub-container or relative to end of previous object

ParameterRef
Parameter that the inclusion of the sub-container depends on

Holds a reference to a parameter that will hold a container id at runtime to expand

element tc:DynamicContainerRefType/DependantOn

Repeating group to include

CountRef
Parameter that contains the count of repeated structures, implies variable length structure

OffsetInBits
Offset in bits from the start of the container scope

Count
0..∞
Repeating group to include

Name                   Type                                  Use                      Default                       Fixed              Annotation
type                      NMTOKENS                                                  Occurs

documentation  Holds a reference to a parameter that will hold a container id at runtime to expand

documentation  Repeating group to include
element **tc:DynamicContainerRefType/OffsetInBits**

**Diagram**

- **tc:BitOffsetType**
  - **AbsoluteOffset**
    - Offset in bits, from the start of this container to start sub-container. Default is 0.
  - **RelativeOffset**
    - Relative offset in bits from end of previous object, can be negative. Default is 0.

**Namespace**

http://www.omg.org/space/tcspec

**Type**

**tc:BitOffsetType**

**Children**

- **AbsoluteOffset**
- **RelativeOffset**

**Annotation** documentation Offset, in bits, from the start of this container to start sub-container. If not present then sub-container start either at offset 0 if the first sub-container or relative to end of previous object.

**Namespace**

http://www.omg.org/space/tcspec

**Type**

**tc:ComparisonCheckType**

**Children**

- **ParameterRef**
- **ComparisonOperator**
- **ArgumentRef**
- **ParameterRef**
- **Value**

**Annotation** documentation Parameter that the inclusion of the sub-container depends on.

**Element** **tc:DynamicContainerRefType/OffsetInBits**

- **DependantOn**
  - Parameter that the inclusion of the sub-container depends on.

**Namespace**

http://www.omg.org/space/tcspec

**Type**

**tc:ComparisonCheckType**

**Children**

- **ParameterRef**
- **ComparisonOperator**
- **ArgumentRef**
- **ParameterRef**
- **Value**

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>calibratedComparison</td>
<td>boolean</td>
<td>false</td>
<td></td>
<td></td>
<td>Should the comparison be based on the calibrated format or not, if true and a constant value is used this should be in calibrated format too.</td>
</tr>
</tbody>
</table>

**Annotation** documentation Parameter that the inclusion of the sub-container depends on.
**element** tc:DynamicContainerRefType/ParameterRef

Diagram:
- **ParameterRef**: Parameter that will hold container id at runtime
- **DependantOn**: Optional parameter set condition that must be true for this parameter to be valid
- **OffsetInBits**: Offset in bits from the start of the container scope, the first bit in the container has an offset of 0

**namespace** http://www.omg.org/space/tcspec
**type** tc:OffsetParameterRefType
**children**
- **Occurs**
- **DependantOn**
- **OffsetInBits**

**attributes**
- **parameterName**: tc:NameReferenceType, required
- **spaceSystemName**: string, optional
- **TimeOffset**: integer, optional
  - Default: 0
- **documentation**: Time delay in milliseconds
- **annotation**: documentation Parameter that will hold container id at runtime

**complexType** tc:MessageKeyMatchListType

Diagram:
- **tc:MessageKeyMatchListType**
- **KeyMatch**

**namespace** http://www.omg.org/space/tcspec
**children**
- **KeyMatch**

**used by**
- **element** tc:ContainerType/ImplementsMessage

**attributes**
- **name**: tc:MessageNameType, required
- **AllMustBeTrue**: boolean

**annotation**
- documentation Holds a list of values for the keys of a message

**element** tc:MessageKeyMatchListType/KeyMatch

Diagram:
- **KeyMatch**

**namespace** http://www.omg.org/space/tcspec
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>tc:MessageKeyIdType</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>Operator</td>
<td>tc:ComparisonOperatorsType</td>
<td></td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>Value</td>
<td>hexBinary</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
</tbody>
</table>

Type: tc:MessageKeyMatchType

Annotation: Value to use for message key match
complexType **tc:MessageKeyMatchType**

```xml
<complexType name="tc:MessageKeyMatchType">
    <documentation>Holds a value to use in a match to a refered to key</documentation>
</complexType>
```

namespace http://www.omg.org/space/tcspec

used by **element** **tc:MessageKeyMatchListType/KeyMatch**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>tc:MessageKeyIdType</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>Operator</td>
<td>tc:ComparisonOperatorsType</td>
<td></td>
<td></td>
<td></td>
<td>Operator to compare with,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>optional, if not present</td>
</tr>
<tr>
<td>Value</td>
<td>hexBinary</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
</tbody>
</table>

**annotation** documentation Holds a value to use in a match to a refered to key

complexType **tc:MessageKeyType**

```xml
<complexType name="tc:MessageKeyType">
    <documentation>Denotes a bit field in a message that can be used to identify a particular message</documentation>
</complexType>
```

namespace http://www.omg.org/space/tcspec

used by **element** **tc:MessageType/Key**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>tc:MessageKeyIdType</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>ShortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td>Optional short key description</td>
</tr>
<tr>
<td>WidthInBits</td>
<td>positiveInteger</td>
<td>required</td>
<td></td>
<td></td>
<td>Width of key field in bits</td>
</tr>
</tbody>
</table>

**annotation** documentation Denotes a bit field in a message that can be used to identify a particular message

**element** **tc:MessageKeyType/LongDescription**

```xml
<element name="LongDescription" type="tc:LongDescriptionType">
    <documentation>Optional long key description</documentation>
</element>
```

namespace http://www.omg.org/space/tcspec

used by **element** **LongDescription**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>tc:MessageKeyIdType</td>
<td></td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
</tbody>
</table>

**annotation** documentation Optional long key description

**element** **tc:MessageKeyType/OffsetInBits**

```xml
<element name="OffsetInBits" type="positionalInteger">
    <documentation>Offset relative to start of container representing this message</documentation>
</element>
```
complexType `tc:MessageType`

```xml
<complexType name="tc:MessageType">
  <element name="LongDescription" type="tc:LongDescriptionType" maxOccurs="0..1" minOccurs="0"/>
  <element name="Key" type="tc:Key"/>
</complexType>
```

**namespace** http://www.omg.org/space/tcspec

**type** `tc:BitOffsetType`

- **AbsoluteOffset**
  - Absolute offset in bits. Default is 0.
- **RelativeOffset**
  - Relative offset in bits from end of previous object, can be negative. Default is 0.

**documentation** Offset relative to start of container representing this message.

**complexType** `tc:MessageType`

```xml
<complexType name="tc:MessageType">
  <element name="LongDescription" type="tc:LongDescriptionType" maxOccurs="0..1" minOccurs="0"/>
  <element name="Key" type="tc:Key"/>
</complexType>
```

**namespace** http://www.omg.org/space/tcspec

**type** `tc:LongDescriptionType`

- **LongDescription**
  - Optional long description of message.
- **Key**
  - Optional set of keys that this identify a message of this type.

**documentation** An abstract object that can be used to help identify structures on an incoming stream.

**element** `tc:MessageType/LongDescription`

```xml
<element name="LongDescription" type="tc:LongDescriptionType" maxOccurs="0..1" minOccurs="0"/>
```

**namespace** http://www.omg.org/space/tcspec

**type** `tc:LongDescriptionType`

- **LongDescription**
  - Optional long description of message.
- **Key**
  - Optional set of keys that this identify a message of this type.

**documentation** An abstract object that can be used to help identify structures on an incoming stream.

**element** `tc:MessageType/Key`

```xml
<element name="Key" type="tc:Key"/>
```

**namespace** http://www.omg.org/space/tcspec

**type** `tc:Key`

- **Key**
  - Optional set of keys that this identify a message of this type.

**documentation** An abstract object that can be used to help identify structures on an incoming stream.
complexType tc:PackagingDefinitionType

namespace http://www.omg.org/space/tcspec
tc:MessageKeyType

LongDescription OffsetInBits

attributes
Id tc:MessageKeyIdType required
ShortDescription tc:ShortDescriptionType optional
WidthInBits positiveInteger required

annotation documentation Optional set of keys that this identify a message of this type

complexType tc:PackagingDefinitionType

diagram

messageList ContainerList NonStandardData

element tc:Packaging
documentation Complex type that holds all the elements of the packaging system

element tc:PackagingDefinitionType/MessageList

diagram

MessageList Message

namespace http://www.omg.org/space/tcspec
Message

key MessageKey Refer Selector Message Field(s) Id

documentation Holds the list of message definitions

element tc:PackagingDefinitionType/MessageList/Message
element **tc:**PackagingDefinitionType/ContainerList

```xml
diagram
  ContainerList --> Container
  ContainerList holds the list of container definitions
```

**Container**

```xml
  Container is a single container definition
```

**attributes**
- **name**
  - Type: tc:MessageNameType
  - Use: required
- **shortDescription**
  - Type: tc:ShortDescriptionType
  - Use: optional

**identity**
- **key**
  - Type: ContainerKey

**annotation**
- **documentation**
  - A single container definition

**namespace**
- http://www.omg.org/space/tcspec

**type**
- tc:MessageType

**children**
- **LongDescription**
  - Key
  - An optional long description of the message

- **Key**
  - Optional set of keys that identify a message of this type

**attributes**
- **name**
  - Type: tc:MessageNameType
  - Use: required

- **shortDescription**
  - Type: tc:ShortDescriptionType
  - Use: optional

**identity**
- **key**
  - ContainerKey

**annotation**
- **documentation**
  - Unique identifier for the message
  - Optional short name of the message

**namespace**
- http://www.omg.org/space/tcspec

**type**
- tc:MessageType

**children**
- **LongDescription**
  - Key
  - An optional long description of the message

- **Key**
  - Optional set of keys that identify a message of this type

**attributes**
- **name**
  - Type: tc:MessageNameType
  - Use: required

- **shortDescription**
  - Type: tc:ShortDescriptionType
  - Use: optional

**identity**
- **key**
  - ContainerKey

**annotation**
- **documentation**
  - A single message definition
element tc:PackagingDefinitionType/NonStandardData

complexType tc:NonStandardDataType

complexType tc:SequenceType
element **tc:SequenceType/Occurs**

Used to describe a sequence of sequences, containers, parameters, or choices

**tc:SequenceType**

- **Occurrences**: Repeating group to include
- **Occurr**: 0..∞

- **ContainerRef**: Set of command arguments to include
- **DynamicContainerRef**: Holds a reference to a parameter that will hold an onboard Id
- **ArgumentRef**: Set of parameters to include
- **ParameterRef**: May also use a reference to a constant parameter
- **DynamicParameterRef**: Holds a reference to a parameter that will hold an onboard Id
- **BinaryConstant**: Holds a reference to a parameter that will hold an onboard Id
- **Choice**: May also use a reference to a constant parameter
- **Sequence**: May also use a reference to a constant parameter

**namespace**: http://www.omg.org/space/tcspec

**children**: Occurs ContainerRef DynamicContainerRef ArgumentRef ParameterRef DynamicParameterRef BinaryConstant Choice Sequence

**used by elements**: tc:ContainerType/Sequence tc:SequenceType/Sequence

**documentation**: Used to describe a sequence of sequences, containers, parameters, or choices
**tc:OccursType**

CountRef | Count | OffsetInBits
--- | --- | ---

**type** | **NMTOKENS** | **Use**
--- | --- | ---

**Default** | **Fixed** | **Annotation**
--- | --- | ---

**Occurs** |  |  

**documentation** | Repeating group to include
element `tc:SequenceType/ContainerRef`

```xml
<element name="ContainerRef" type="tc:ContainerRefType">
  <Occurs minOccurs="0" maxOccurs="999999999"/>
  <DependantOn/>
  <OffsetInBits/>
</element>
```

**attributes**
- `nameRef`: `tc:ContainerNameType` (required)
- `Id of container to contain`

**namespace**
- `http://www.omg.org/space/tcspec`

---

element `tc:SequenceType/DynamicContainerRef`

```xml
<element name="DynamicContainerRef" type="tc:DynamicContainerRefType">
  <Occurs minOccurs="0" maxOccurs="999999999"/>
  <DependantOn/>
  <OffsetInBits/>
  <ParameterRef/>
</element>
```

**attributes**
- `nameRef`: `tc:ContainerNameType` (required)
- `Id of container to contain`

**namespace**
- `http://www.omg.org/space/tcspec`
type tc:DynamicContainerRefType

Children: Occurs, DependantOn, OffsetInBits, ParameterRef
element tc:SequenceType/ArgumentRef

```xml
<tc:SequenceType/ArgumentRef>
  <tc:OffsetParameterRefType>
    <Occurs>0,∞</Occurs>
    <DependantOn/>
    <OffsetInBits>Offset in bits from the start of the container scope, the first bit in the container has an offset of 0</OffsetInBits>
  </tc:OffsetParameterRefType>
  <ArgumentRef>Set of command arguments to include</ArgumentRef>
</tc:SequenceType/ArgumentRef>
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterName</td>
<td>tc:NameReferenceType</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>spaceSystemName</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td>Time delay in milliseconds</td>
</tr>
<tr>
<td>TimeOffset</td>
<td>integer</td>
<td>optional</td>
<td>0</td>
<td></td>
<td>documentation</td>
</tr>
</tbody>
</table>
```

```
annotation documentation Set of command arguments to include
```

element tc:SequenceType/ParameterRef

```xml
<tc:SequenceType/ParameterRef>
  <tc:OffsetParameterRefType>
    <Occurs>0,∞</Occurs>
    <DependantOn/>
    <OffsetInBits>Offset in bits from the start of the container scope, the first bit in the container has an offset of 0</OffsetInBits>
  </tc:OffsetParameterRefType>
  <ParameterRef>Set of parameters to include</ParameterRef>
</tc:SequenceType/ParameterRef>
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterName</td>
<td>tc:NameReferenceType</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>spaceSystemName</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
annotation documentation
```

namespace http://www.omg.org/space/tcspec
tc:OffsetParameterRefType
Occurs DependantOn OffsetInBits
<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>parameterName</td>
<td>tc:NameReferenceType</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>spaceSystemName</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TimeOffset</td>
<td>integer</td>
<td>optional</td>
<td>0</td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td>annotation</td>
<td>documentation</td>
<td>Set of parameters to include</td>
<td></td>
<td></td>
<td></td>
<td>Time delay in milliseconds</td>
</tr>
</tbody>
</table>
element tc:SequenceType/DynamicParameterRef

diagram

namespace http://www.omg.org/space/tcspec
tc:OffsetParameterRefType

Occurs DependantOn OffsetInBits

attributes
Name                   Type                                  Use                      Default                       Fixed              Annotation
parameterName   tc:NameReferenceType     required
spaceSystemName  string                               optional               0                                      documentation
TimeOffset           integer                               optional               0                                      documentation

annotation
documentation  Holds a reference to a parameter that will hold an onboard Id

Time delay in milliseconds

element tc:SequenceType/BinaryConstant

diagram

namespace http://www.omg.org/space/tcspec
LongDescription OffsetInBits

attributes
lengthInBits          nonNegativeInteger           required
value                   hexBinary                          required

annotation
documentation  May also use a reference to a constant parameter

element tc:SequenceType/BinaryConstant/LongDescription

diagram

namespace http://www.omg.org/space/tcspec
tc:LongDescriptionType

annotation
documentation  Optional long description of fixed area
element tc:SequenceType/BinaryConstant/OffsetInBits

diagram

namespace http://www.omg.org/space/tcspec
type tc:BitOffsetType
children AbsoluteOffset RelativeOffset

element tc:SequenceType/Choice

diagram

namespace http://www.omg.org/space/tcspec
type tc:ChoiceType
children Occurs ContainerRef

element tc:SequenceType/Sequence
complexType tc:ServiceType

namespace http://www.omg.org/space/tcspec
tc:SequenceType

Occurs
ContainerRef
DynamicContainerRef
ArgumentRef
ParameterRef
DynamicParameterRef
BinaryConstant
Choice
Sequence

LongDescription
Optional long description of this service

tc:ServiceType

Holds a set of services, logical groups of messages and containers

MessageList

Set of messages this service uses

ContainerList

Set of containers this service uses
<table>
<thead>
<tr>
<th>used by</th>
<th>element</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>name</td>
<td>tc:ServiceNameType</td>
<td>required</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
<tr>
<td></td>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td>optional</td>
<td></td>
<td></td>
<td>documentation</td>
</tr>
</tbody>
</table>

**annotation documentation** Holds a set of services, logical groups of messages and containers
element **tc:ServiceType/LongDescription**

```xml
<diagram>
  <LongDescription/>
  <annotation>Optional long description of this service</annotation>
</diagram>
```

**namespace** http://www.omg.org/space/tcspec

**type** tc:LongDescriptionType

**annotation** documentation Optional long description of this service

---

element **tc:ServiceType/MessageList**

```xml
<diagram>
  <MessageList />
  <NameRef />
  <annotation>Set of messages this service uses
  <annotation>reference to an existing message</annotation>
</diagram>
```

**namespace** http://www.omg.org/space/tcspec

**children** NameRef

**annotation** documentation Set of messages this service uses

---

element **tc:ServiceType/MessageList/NameRef**

```xml
<diagram>
  <NameRef />
  <annotation>reference to an existing message</annotation>
</diagram>
```

**namespace** http://www.omg.org/space/tcspec

**type** tc:NameReferenceType

**facets** pattern [a-zA-Z0-9_\-\/]*

**annotation** documentation reference to an existing message

---

element **tc:ServiceType/ContainerList**

```xml
<diagram>
  <ContainerList />
  <Id />
  <annotation>Set of containers this service uses
  <annotation>Id of an existing container</annotation>
</diagram>
```

**namespace** http://www.omg.org/space/tcspec

**children** Id

**annotation** documentation Set of containers this service uses

---

element **tc:ServiceType/ContainerList/Id**

```xml
<Id />
```

**namespace** http://www.omg.org/space/tcspec

**type** tc:ContainerNameType

---
facets pattern [a-zA-Z0-9_\-]*
annotation documentation Id of an existing container
simpleType tc:ContainerNameType

namespace http://www.omg.org/space/tcspec
tc:NameType
tc:ServiceType/ContainerList/Id
tc:ContainerType/@name
tc:ContainerRefType/@nameRef

attributes pattern [a-zA-Z0-9_\-\ .]+*
facets
annotation
documentation Identifier for a container

simpleType tc:MessageKeyIdType

namespace http://www.omg.org/space/tcspec
tc:NameType
tc:MessageKeyType/@Id
tc:MessageKeyMatchType/@Id

attributes pattern [a-zA-Z0-9_\-\ .]+*
facets
annotation
documentation Identifier for a message key

simpleType tc:MessageNameType

namespace http://www.omg.org/space/tcspec
tc:NameType
tc:MessageType/@name
tc:MessageKeyMatchListType/@name

attributes pattern [a-zA-Z0-9_\-\ .]+*
facets
annotation
documentation Identifier for a message

simpleType tc:ServiceNameType

namespace http://www.omg.org/space/tcspec
tc:NameType
tc:ServiceType/@name

attributes pattern [a-zA-Z0-9_\-\ .]+*
facets
annotation
documentation Identifier for a service

complexType tc:ConvolutionalStreamType
The type definition for convolutionally encoded streams. The processing algorithm for the convolutional encoder is described using a series of generator polynomials. Each polynomial corresponds to a modulo two adder. The order of the polynomials corresponds to the order in the output stage. Each term in each polynomial corresponds to a register in the shift register. The coefficients in these polynomials are either "1" for connected or "0" for not connected.

**element** `tc:ConvolutionalStreamType/Polynomial`  

A term in a polynomial expression.

**complexType** `tc:EncryptedStreamType`  
The top level type definition for all encrypted streams.
namespace http://www.omg.org/space/tcspec
extension of tc:PCMStreamType

LongDescription

used by
element tc:StreamType/EncryptedStream

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>tc:NameType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bitRateInBPS</td>
<td>double</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pcmType</td>
<td>string</td>
<td>required</td>
<td>NRZL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inverted</td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>algorithmName</td>
<td>string</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

annotation
documentation The top level type definition for all encrypted streams.
complexType **tc:FixedFrameStreamType**

namespace http://www.omg.org/space/tcspec
type extension of tc:PCMStreamType

diagram

LongDescription SyncStrategy
clements tc:FixedFrameSync/FixedFrameStream tc:StreamType/FixedFrameStream

children

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>tc:NameType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bitRateInBPS</td>
<td>double</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pcmType</td>
<td>string</td>
<td>NRZL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inverted</td>
<td>boolean</td>
<td>false</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>frameLengthInBits</td>
<td>long</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

annotation
documentation The top level type definition for all data streams with a fixed frame length.

element **tc:FixedFrameStreamType/SyncStrategy**

diagram

type tc:SyncStrategyType

children

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>verifyToLockGoodFrames</td>
<td>integer</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>checkToLockGoodFrames</td>
<td>integer</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxErrorsInSyncWord</td>
<td>integer</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>syncAperature</td>
<td>integer</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

complexType **tc:PCMStreamType**
A PCM Stream Type is the high level definition for all Pulse Code Modulated (PCM) (i.e., binary) streams.

A container Type used to contain a single reference to all known Stream types.

A container Type used to contain a single reference to all known Stream types.
**Diagram: Fixed Frame Stream**

- **Type:** tc:FixedFrameStreamType
- **Attributes:**
  - Name: tc:NameType
  - Short Description: tc:ShortDescriptionType
  - Bit Rate In BPS: double
  - PCM Type: string (NRZL)
  - Inverted: boolean (false)
  - Frame Length In Bits: long (required)

**Diagram: Encrypted Stream**

- **Type:** tc:EncryptedStreamType
- **Attributes:**
  - Name: tc:NameType
  - Short Description: tc:ShortDescriptionType
  - Bit Rate In BPS: double
  - PCM Type: string (NRZL)
  - Inverted: boolean (false)
  - Algorithm Name: string (required)
  - Key: string

**Diagram: Variable Frame Stream**

- **Type:** tc:VariableFrameStreamType
- **Attributes:**
  - Name: tc:NameType
  - Short Description: tc:ShortDescriptionType
  - Bit Rate In BPS: double
  - PCM Type: string (NRZL)
  - Inverted: boolean (false)
  - Zero Bit Insertion: boolean (true)
  - Key: string
element `tc:StreamType/ConvolutionalStream`

```xml
<complexType name="ConvolutionalStreamType">
  <sequence>
    <element name="LongDescription" type="tc:LongDescriptionType"/>
    <element name="Polynomial" minOccurs="2..inf" maxOccurs="1" type="tc:Polynomial"/>
  </sequence>
</complexType>
```

**attributes**
- **name** (`tc:NameType`)
- **shortDescription** (`tc:ShortDescriptionType`)
- **bitRateInBP** (`double`)
- **pcmType** (`string`) default: NRZL
- **inverted** (`boolean`) default: false

element `tc:StreamType/NonStandardData`

```xml
<complexType name="NonStandardDataType">
  <sequence>
    <element name="any" minOccurs="0" maxOccurs="inf" type="anyType"/>
  </sequence>
</complexType>
```

**attributes**
- **usage** (`string`) default: required

complexType `tc:SyncStrategyType`

A Sync Strategy specifies the requirements to deem a PCM Fixed Frame Stream "in-sync" or out of sync.

```xml
<complexType name="SyncStrategyType">
  <sequence>
    <element name="AutoInvert" type="tc:AutoInvert"/>
    <element name="MessageMatch" minOccurs="0" maxOccurs="inf" type="tc:MessageMatch"/>
    <element name="SyncPattern" minOccurs="0" maxOccurs="inf" type="tc:SyncPattern"/>
  </sequence>
</complexType>
```

**namespace** http://www.omg.org/space/tcspec

**used by**
- `tc:FixedFrameStreamType/SyncStrategy` `tc:VariableFrameStreamType/SyncStrategy`
<table>
<thead>
<tr>
<th>attributes</th>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>verifyToLockGoodFrames</td>
<td>integer</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>checkToLockGoodFrames</td>
<td>integer</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>maxErrorsInSyncWord</td>
<td>integer</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>syncAperature</td>
<td>integer</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Annotation**

A Sync Strategy specifies the requirements to deem a PCM Fixed Frame Stream "in-sync" or out of sync.
element `tc:SyncStrategyType/AutoInvert`

```xml
diagram
  AutoInvert
  After a certain number of bad frames, it may be desirable to invert the incoming data, and then look for frame sync.
attributes
  Name: badFramesToAutoInvert, Type: positiveInteger, Use: , Default: 2, Fixed: , Annotation: documentation After a certain number of bad frames, it may be desirable to invert the incoming data, and then look for frame sync.
```

element `tc:SyncStrategyType/MessageMatch`

```xml
diagram
  MessageMatch
  NameRef
  Reference to an existing message
children
  NameRef
attributes
  Name: bitLocation, Type: integer, Use: , Default: 0, Fixed: , Annotation: documentation Reference to an existing message
```

element `tc:SyncStrategyType/MessageMatch/NameRef`

```xml
NameRef
  Reference to an existing message
type
  tc:NameReferenceType
facets
  pattern [a-zA-Z0-9 \._/]*
annotation
documentation Reference to an existing message
```

element `tc:SyncStrategyType/SyncPattern`

```xml
diagram
  SyncPattern
  The pattern of bits used to look for frame synchronization.
attributes
  Name: pattern, Type: hexBinary, Use: required, Default: , Fixed: , Annotation: documentation The pattern of bits used to look for frame synchronization.
```
complexType tc:VariableFrameStreamType

namespace http://www.omg.org/space/tcspec

type extension of tc:PCMStreamType

LongDescription SyncStrategy

element tc:StreamType/VariableFrameStream

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>tc:NameType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shortDescription</td>
<td>tc:ShortDescriptionType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bitRateInBPS</td>
<td>double</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pcmType</td>
<td>string</td>
<td></td>
<td>NRZL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inverted</td>
<td>boolean</td>
<td></td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>zeroBitInsertion</td>
<td>boolean</td>
<td>optional</td>
<td>true</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

documentation The top level type definition for all data streams with a variable frame length.

element tc:VariableFrameStreamType/SyncStrategy

namespace http://www.omg.org/space/tcspec

type restriction of string

used by attribute tc:StreamType/@streamSource

complexType tc:SyncStrategyType

AutoInvert MessageMatch SyncPattern

attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>verifyToLockGoodFrames</td>
<td>integer</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>checkToLockGoodFrames</td>
<td>integer</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxErrorsInSyncWord</td>
<td>integer</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>syncAperature</td>
<td>integer</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

documentation After a certain number of bad frames, it may be desirable to invert the incoming data, and then look for frame sync.

documentation The pattern of bits used to look for frame synchronisation.

complexType tc:StreamSourceType

namespace http://www.omg.org/space/tcspec

restriction of string

used by attribute tc:StreamType/@streamSource
facets
  enumeration TC
  enumeration TM
APPENDIX A – XML SCHEMA

SpaceSystem.xsd

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Nori Jungmann (Engineering) -->

Style Notes, used throughout all schemas:
- Element and Type names begin with a capital letter.
- Type names end with the word "Type".
- Attribute names begin with a lowercase letter.
- Usually, when the UML class diagram references classes, W3C Elements are used, and whenever the UML refeneces simple types (strings, ints), W3C Attributes are used. In general, attributes are preferred over elements because they're easier to deal with in SAX and DOM, but whenever the Element/Attribute may one day carry metadata, elements should be used. One exception, is enumerated classes, because enumerations may be defined for attributes but not for elements.
- Bias toward self-describing names over short, bandwidth conserving ones.
- Use mixed case in names rather than underscores to combine multiple words (camelCase).
- A documentation annotation is included in every element and type definition.
- Hints on units (for values with units) are provided in the names of attributes and elements (e.g. "dataRateInBPS" is preferred over "dataRate" OR "frameLengthInBits" is preferred over "frameLength").
- Major elements or any elements used multiple times are first defined with a complexType definition.

-->
<element name="CommandDefinition" type="tc:CommandDefinitionType" maxOccurs="unbounded"/>
</complexType>
<key name="CommandDefinitionListKey">
<selector xpath="/CommandDefinition"/>
<field xpath="@name"/>
</key>
</element>
<element name="ParameterList" minOccurs="0">
<complexType>
<sequence>
<element name="Parameter" type="tc:ParameterType" maxOccurs="unbounded">
<annotation>
<documentation xml:lang="en">A single parameter definition</documentation>
</annotation>
</element>
</sequence>
</complexType>
<key name="ParameterListKey">
<selector xpath="/Parameter"/>
<field xpath="@name"/>
</key>
</element>
<element ref="tc:Packaging" minOccurs="0">
<complexType>
<documentation>Optional description of message</documentation>
</complexType>
</element>
<element name="PropertyList">
<complexType>
<sequence>
<element name="Property" type="tc:PropertyType" maxOccurs="unbounded">
<annotation>
<documentation xml:lang="en">Using these properties is discouraged, they are only meant as a
placeholder for data that does not otherwise fit into this schema.</documentation>
</annotation>
</element>
</sequence>
</complexType>
</element>
<element name="ServiceList">
<complexType>
<sequence>
<element name="Service" type="tc:ServiceType" maxOccurs="unbounded"/>
</sequence>
</complexType>
<key name="ServiceListKey">
<selector xpath="/Service"/>
<field xpath="@name"/>
</key>
</element>
<element name="AlgorithmList">
<complexType>
<choice maxOccurs="unbounded">
<element name="CustomAlgorithm" type="tc:CustomAlgorithmType"/>
<element name="ConcatenationAlgorithm" type="tc:ConcatenationAlgorithmType"/>
<element name="MathAlgorithm" type="tc:MathAlgorithmType"/>
<element name="ReedSolomonEncoder" type="tc:ReedSolomonType"/>
<element name="ReedSolomonDecoder" type="tc:ReedSolomonType"/>
</choice>
</complexType>
</element>
<element name="StreamList">
<complexType>
<sequence>
<element name="Stream" type="tc:StreamType" maxOccurs="unbounded"/>
</sequence>
</complexType>
</element>
<element name="NonStandardData" type="tc:NonStandardDataType" minOccurs="0"/>
Parameter.xsd

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v5 rel. 2 U (http://www.xmlspy.com) by Ed Shaya (NASA) -->
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Nori Jungmann (Engineering) -->
  <annotation>
    <documentation>$Id: Parameter.xsd,v 1.18 2003/03/03 14:34:40 gerry Exp $</documentation>
    <documentation xml:lang="en">This schema defines the structure for a Parameter. A Parameter is an object which can assume a value (parameter instance). </documentation>
  </annotation>
  <include schemaLocation="CommonTypes.xsd"/>
  <complexType name="ParameterType">
    <annotation>
      <documentation xml:lang="en">A type definition used by Parameter</documentation>
    </annotation>
    <complexContent>
      <extension base="tc:BaseParameterType">
        <sequence>
          <element name="Alias" type="tc:AliasType" minOccurs="0" maxOccurs="unbounded"/>
          <element name="SystemName" type="tc:SystemNameType" minOccurs="0">
            <annotation>
              <documentation>Optional. Normally used when the database is built in a flat, non-hierarchical format</documentation>
            </annotation>
          </element>
          <element name="ToString" type="tc:ToStringType" minOccurs="0"/>
          <element name="Validity" type="tc:ComparisonCheckType" minOccurs="0"/>
          <choice>
            <documentation>The database builder is allowed to specify alarms as either simple critical/warning style alarms or (almost) arbitrarily complex alarms using alarm conditions</documentation>
            <element name="AlarmConditions" type="tc:ComplexAlarmsType" minOccurs="0" maxOccurs="unbounded"/>
            <element name="Alarms" type="tc:SimpleAlarmsType"/>
          </choice>
          <element name="PhysicalAddress" minOccurs="0" maxOccurs="unbounded">
            <annotation>
              <documentation>Contains the address (channel information) required to process the spacecraft telemetry streams</documentation>
            </annotation>
            <complexType>
              <complexContent>
                <extension base="tc:PhysicalAddressType"/>
              </complexContent>
            </complexType>
          </element>
          <element name="NonStandardData" type="tc:NonStandardDataType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="dataSource" use="optional">
          <simpleType>
            <restriction base="string">
              <enumeration value="telemetered"/>
              <enumeration value="derived"/>
              <enumeration value="constant"/>
              <enumeration value="local"/>
            </restriction>
          </simpleType>
        </attribute>
        <attribute name="readOnly" type="boolean" use="optional" default="false"/>
        <attribute name="validRangeAppliesToCalibrated" type="boolean" use="optional" default="true"/>
        <attribute name="onboardID" type="string" use="optional"/>
      </extension>
    </complexContent>
  </complexType>
</schema>
<complexType name="ParameterRefMatchListType">
    <annotation>
        <documentation xml:lang="en">Holds a list of parameter matches, and whether all or one match required</documentation>
    </annotation>
    <sequence>
        <element name="ParameterMatch" type="tc:ComparisonCheckType" maxOccurs="unbounded">
            <annotation>
                <documentation xml:lang="en">Condition that must be evaluated</documentation>
            </annotation>
        </element>
    </sequence>
    <attribute name="AllMustBeTrue" type="boolean">
        <annotation>
            <documentation xml:lang="en">Must all matched be true? If not present then all must be true</documentation>
        </annotation>
    </attribute>
</complexType>

<complexType name="ParameterRefType">
    <annotation>
        <documentation xml:lang="en">A reference to a Parameter that includes a validity check for the parameters</documentation>
    </annotation>
    <complexContent>
        <extension base="tc:SimpleParameterRefType">
            <sequence>
                <element name="Occurs" type="tc:OccursType" minOccurs="0" maxOccurs="unbounded">
                    <annotation>
                        <documentation xml:lang="en">Repeating group to include</documentation>
                    </annotation>
                </element>
                <element name="DependantOn" type="tc:ParameterRefMatchListType" minOccurs="0">
                    <annotation>
                        <documentation xml:lang="en">Optional parameter set condition that must be true for this parameter to be valid</documentation>
                    </annotation>
                </element>
            </sequence>
            <attribute name="TimeOffset" type="integer" use="optional" default="0">
                <annotation>
                    <documentation xml:lang="en">Time delay in milliseconds</documentation>
                </annotation>
            </attribute>
        </extension>
    </complexContent>
</complexType>

<complexType name="OffsetParameterRefType">
    <annotation>
        <documentation xml:lang="en">Holds an offset to a referred parameter and optional comparisons to determine validity of this parameter</documentation>
    </annotation>
    <complexContent>
        <extension base="tc:ParameterRefType">
            <sequence>
                <element name="OffsetInBits" type="tc:BitOffsetType" minOccurs="0">
                    <annotation>
                        <documentation xml:lang="en">Offset in bits from the start of the container scope, the first bit in the container has an offset of 0</documentation>
                    </annotation>
                </element>
            </sequence>
            <attribute name="TimeOffset" type="integer" use="optional" default="0">
                <annotation>
                    <documentation xml:lang="en">Time delay in milliseconds</documentation>
                </annotation>
            </attribute>
        </extension>
    </complexContent>
</complexType>

<complexType name="ComplexAlarmsType">
    <annotation>
        <documentation>Alarms are used to notify the operator when a parameter value is outside of acceptable ranges or states. Dynamic or rate-of-change Limit Ranges could be considered.</documentation>
    </annotation>
    <choice maxOccurs="unbounded">
        <element name="StaticRanges" type="tc:RangeListType">
            <annotation>
                <documentation xml:lang="en">A StaticRange has an optional UseRangeWhen condition and one or more AlarmRanges</documentation>
            </annotation>
        </element>
    </choice>
</complexType>
<element name="RateOfChangeRanges">
  <annotation>
    <documentation>Like a Static Range, but it is based on the rate of change for the Parameter.</documentation>
  </annotation>
  <complexType>
    <complexContent>
      <extension base="tc:RangeListType">
        <attribute name="perTimeBase" use="optional" default="second">
          <annotation>
            <documentation xml:lang="en">Rate of change alarms must be reference to some relative time base e.g. 3.4 volts per second</documentation>
          </annotation>
          <simpleType>
            <restriction base="string">
              <enumeration value="microsecond"/>
              <enumeration value="millisecond"/>
              <enumeration value="second"/>
            </restriction>
          </simpleType>
        </attribute>
      </extension>
    </complexContent>
  </complexType>
</element>

<element name="States">
  <annotation>
    <documentation>Contains multiple (un)acceptable values for multi-state parameters.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="UseRangeWhen" minOccurs="0">
        <annotation>
          <documentation>Can be used to implement conditional limits</documentation>
        </annotation>
        <complexType>
          <choice>
            <element name="EqualityCheck" type="tc:ComparisonCheckType"/>
            <element name="CustomCheck"/>
          </choice>
        </complexType>
      </element>
      <element name="State" maxOccurs="unbounded">
        <annotation>
          <documentation xml:lang="en">(Un)acceptable values for multi-state parameters.</documentation>
        </annotation>
        <complexType>
          <attribute name="statePriority" type="nonNegativeInteger" use="optional"/>
          <attribute name="type" type="tc:LimitCheckType" use="required"/>
          <attribute name="state" type="nonNegativeInteger" use="required"/>
        </complexType>
      </element>
    </sequence>
    <attribute name="limitSetPriority" type="nonNegativeInteger" use="optional"/>
  </complexType>
</element>

<element name="CustomAlarmCheck" type="tc:CustomAlgorithmType">
  <annotation>
    <documentation>An external algorithm that can be arbitrarily complex. E.g. an alarm that occurs when the parameter's FFT shows a resonance at 20Hz.</documentation>
  </annotation>
</element>

<attribute name="minViolations" use="required">
  <annotation>
    <documentation xml:lang="en">Doesn't get reported until it has been out of range for this value times</documentation>
  </annotation>
  <restriction base="integer">
    <minInclusive value="0"/>
    <maxInclusive value="99"/>
  </restriction>
</attribute>
<complexType name="DwellSet">
    <annotation>
        <documentation xml:lang="en">Dwelling parameters</documentation>
    </annotation>
    <complexType>
        <attribute name="onboardID" type="string" use="required"/>
        <attribute name="address" type="string" use="required"/>
        <attribute name="startBit" type="nonNegativeInteger" use="required"/>
        <attribute name="length" type="nonNegativeInteger" use="required"/>
        <attribute name="ParameterName" type="tc:NameReferenceType" use="required"/>
        <attribute name="alternateParameterName" type="tc:NameReferenceType" use="optional"/>
    </complexType>
</element>

<complexType name="VariableParameterType">
    <annotation>
        A special parameter that may have a variable width
    </annotation>
    <complexContent>
        <extension base="tc:ParameterType">
            <choice>
                <element name="WidthRef" type="tc:ParameterRefType"/>
                <element name="stop" type="tc:StopType"/>
            </choice>
        </extension>
    </complexContent>
</complexType>

<complexType name="StopType">
    <annotation>
        Specifies either a width in bits or a termination for variable width parameters
    </annotation>
    <attribute name="representation" type="hexBinary"/>
    <attribute name="widthInBits" type="positiveInteger"/>
</complexType>

<complexType name="SimpleAlarmsType">
    <annotation>
        A simple warning, critical alarm type
    </annotation>
    <sequence>
        <element name="CriticalRange" type="tc:RangeType" minOccurs="0"/>
        <element name="WarningRange" type="tc:RangeType" minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="RangeListType">
    <annotation>
        Contains multiple ranges
    </annotation>
    <sequence>
        <element name="UseRangeWhen" minOccurs="0"/>
        <element name="AlarmRange" maxOccurs="unbounded">
            <complexType>
                <complexContent>
                    <extension base="tc:RangeType">
                        <attribute name="severity" type="nonNegativeInteger" use="required"/>
                        <attribute name="type" type="tc:LimitCheckType" use="required"/>
                        <attribute name="rangeName" type="string" use="optional"/>
                    </extension>
                </complexContent>
            </complexType>
        </element>
    </sequence>
</complexType>
When it's important to know the physical address(s) on the spacecraft that this parameter may be collected from, use this.

```xml
<complexType name="PhysicalAddressType">
    <annotation>
        <documentation>When it's important to know the physical address(s) on the spacecraft that this parameter may be collected from, use this.</documentation>
    </annotation>
    <sequence>
        <element name="SubAddress" type="tc:PhysicalAddressType" minOccurs="0"/>
    </sequence>
    <attribute name="sourceName" type="string"/>
    <attribute name="sourceAddress" type="string"/>
</complexType>
</schema>
```
CommandDefinition.xsd

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Nori Jungmann (Engineering) -->

--> $Id: CommandDefinition.xsd,v 1.12 2003/03/03 14:34:40 gerry Exp $

<complexType name="ArgumentType">
  <documentation>A type definition used as the base type for a CommandDefinition and for BusCommandDefinition.</documentation>
  <complexContent>
    <extension base="tc:BaseParameterType">
      <sequence>
        <element name="Argument" type="tc:ArgumentType" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="CommandDefinitionType">
  <documentation xml:lang="en">A type definition used as the base type for a CommandDefinition</documentation>
  <sequence>
    <element name="LongDescription" type="tc:LongDescriptionType" minOccurs="0"/>
    <element name="Alias" type="tc:AliasType" minOccurs="0" maxOccurs="unbounded"/>
    <element name="SystemName" type="tc:SystemNameType" minOccurs="0"/>
    <element name="Interlock" minOccurs="0">
      <documentation>Specifies the interlock for this command, including the scope of the interlock. An interlock restricts commanding ability at the specified scope until this command has reached the specified stage.</documentation>
      <complexType>
        <attribute name="scope" type="tc:InterlockScopeType" use="required"/>
        <attribute name="stage" type="tc:StageType" use="required"/>
      </complexType>
    </element>
    <element name="Field" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

This schema defines the structure for a CommandDefinition. A CommandDefinition provides a description of the command, a name for the command, the allowable arguments for the command, and (optionally) for BusCommands (commands destined for an off-platform bus) the opcode for the command and argument codes.
The binary transform tells the system how to construct the command (transform it to binary format).

A command processor is the name of some external algorithm that is used to format the command. Command processors may also perform additional formatting of the command - formatting that cannot be adequately described in the database.

A command validator is used to validate that a particular user has the proper permissions to issue the command. Command Validators - like command processors - are the name of some external algorithm that does the command validation.

Sets the values of parameters after the command has been verified.

A command constraint is used to check that the command can be run in the current device operating mode or environment. Command constraints are the name of some external algorithm that does the command constraint check or a list of simple comparison checks.
A command verifier is used to check that the command has been successfully executed. Command Verifiers are the name of some external algorithm that does the command verification.

```xml
<sequence>
    <choice>
        <element name="CustomAlgorithm" type="tc:CustomAlgorithmType"/>
        <element name="Comparison" type="tc:ComparisonCheckType" maxOccurs="unbounded"/>
        <element name="ContainerRef" type="tc:ContainerRefType" maxOccurs="unbounded"/>
    </choice>
    <element name="Window">
        <documentation>Specifies how much of a chance to provide for the verification. The window may be specified in time or in number of samples of the comparison parameter to wait for a change.</documentation>
        <complexType>
            <choice>
                <sequence>
                    <element name="StartTime" type="nonNegativeInteger"/>
                    <element name="Interval" type="positiveInteger"/>
                </sequence>
                <element name="ChangeCount" type="positiveInteger"/>
            </choice>
        </complexType>
    </element>
</sequence>
```

```xml
<element name="InterlockScopeType">
    <annotation>
        <documentation xml:lang="en">Specifies the scope of the interlock</documentation>
    </annotation>
    <restriction base="string">
        <enumeration value="SystemGlobal"/>
        <enumeration value="SystemLocal"/>
        <enumeration value="SubsystemGlobal"/>
        <enumeration value="SubsystemLocal"/>
    </restriction>
</element>
```

```xml
<element name="StageType">
    <annotation>
        <documentation xml:lang="en">Specifies the stage of command uplink</documentation>
    </annotation>
    <restriction base="string">
        <enumeration value="Reception"/>
        <enumeration value="Uplink"/>
        <enumeration value="Transfer"/>
        <enumeration value="Acceptance"/>
        <enumeration value="Execution"/>
        <enumeration value="Completion"/>
    </restriction>
</element>
```
Algorithm.xsd

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Ed Shaya (NASA) -->
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Nori Jungmann (Engineering) -->
  <include schemaLocation="CommonTypes.xsd"/>
  <include schemaLocation="StreamDefinitions.xsd"/>
  <include schemaLocation="Parameter.xsd"/>
  <annotation>
    <documentation>$Id: Algorithm.xsd,v 1.13 2003/03/03 07:51:22 cooper Exp $</documentation>
    <documentation xml:lang="en">This schema defines the structure for an Algorithm. An Algorithm may be one of a growing set of pre-defined algorithms or a named escape into a user defined algorithm where (depending on the system) the name of the algorithm may be a java class, a function in a shared library, an external program or some other reference to an outside algorithm.
    </documentation>
  </annotation>
  <complexType name="CustomAlgorithmType">
    <annotation>
      <documentation xml:lang="en">The type definition used by the Custom Algorithm element.</documentation>
      <documentation xml:lang="en">Complete algorithm written in the algorithm language. Embedded new line characters are legal within this attribute.</documentation>
    </annotation>
    <sequence>
      <element name="LongDescription" type="tc:LongDescriptionType" minOccurs="0"/>
      <element name="Trigger" type="tc:TriggerType" minOccurs="0"/>
      <element name="Inputs" minOccurs="0" maxOccurs="unbounded">
        <complexType>
          <choice maxOccurs="unbounded">
            <element name="ParameterRef">
              <annotation>
                <documentation>Names an input parameter to the algorithm. There are two attributes to InputParm, inputName and parameterName. parameterName is a parameter reference name for a parameter that will be used in this algorithm. inputName is an optional “friendly” name for the input parameter.</documentation>
              </annotation>
              <complexType>
                <complexContent>
                  <extension base="tc:ParameterRefType">
                    <attribute name="inputName" type="string"/>
                  </extension>
                </complexContent>
              </complexType>
            </element>
            <element name="Constant" minOccurs="0">
              <annotation>
                <documentation xml:lang="en">Names and provides a value for a constant input to the algorithm. There are two attributes to Constant, constantName and value. constantName is a variable name in the algorithm to be executed. value is the value of the constant to be used.</documentation>
              </annotation>
              <complexType>
                <complexContent>
                  <attribute name="constantName" type="string"/>
                  <attribute name="value" type="string" use="required"/>
                </complexContent>
              </complexType>
            </element>
          </choice>
        </complexType>
        <element name="OutputParameterRef" minOccurs="0" maxOccurs="unbounded">
          <annotation>
            <documentation>Names an output parameter to the algorithm. There are two attributes to OutputParm, outputName and parameterName. parameterName is a parameter reference name for a parameter that will be updated by this algorithm. outputName is an optional “friendly” name for the output parameter.</documentation>
          </annotation>
          <complexType>
            <complexContent>
              <extension base="tc:ParameterRefType">
                <attribute name="outputName" type="string"/>
              </extension>
            </complexContent>
          </complexType>
        </element>
      </element>
    </sequence>
  </complexType>
</schema>
```

At some later date, this schema may also allow the logic of the user defined algorithm to be defined within the instance document itself (perhaps using MathML?).

All algorithms should contain a change log that is modified with each update.
This optional element may be used to enter Pseudo or actual code for the algorithm. The language for the algorithm is specified with the language attribute.

This is the external algorithm. Multiple entries are provided so that the same database may be used for multiple implementations.

First telemetry container from which the output parameter should be calculated.

Algorithm processing priority.

Calibrators are normally used to convert to and from bit compacted numerical data.

A calibration type where a segmented line in a raw vs calibrated plane is described using a set of points. Raw values are converted to calibrated values by finding a position on the line corresponding to the raw value. The algorithm triggers on the input parameter.

A calibration type where a curved in a raw vs calibrated plane is described using a set of polynomial coefficients. Raw values are converted to calibrated values by finding a position on the curve corresponding to the raw value. The first coefficient belongs with the X^0 term, the next coefficient belongs to the X^1 term and so on. The algorithm triggers on the condition parameter.

A calibration type where the bus value is converted to a string value.
<complexType name="FixedFrameSync">
  <annotation>
    <documentation xml:lang="en">An algorithm most often implemented in hardware, that finds the frame synchronization mark in each frame. Given raw binary data, a PCM Fixed Frame Sync will output framed data. Triggers on the input parameter</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="InputParameterRef" type="tc:ParameterRefType"/>
      <element name="OutputParameterRef" type="tc:ParameterRefType"/>
    </sequence>
  </complexType>
</complexType>

<complexType name="ConcatenationAlgorithmType">
  <annotation>
    <documentation>An algorithm that will concatenate two or more parameters. Although not checked by the schema, input parameters must be binary or boolean and the resulting output parameter must be sized to contain the concatenation.</documentation>
  </annotation>
  <sequence>
    <element name="InputParameterList">
      <complexType>
        <sequence>
          <element name="InputParameterRef" type="tc:ParameterRefType" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="OutputParameterRef" type="tc:ParameterRefType"/>
  </sequence>
</complexType>

<complexType name="MathAlgorithmType">
  <annotation>
    <documentation>A simple mathematical operation</documentation>
  </annotation>
  <complexContent>
    <extension base="tc:MathOperationType">
      <sequence>
        <element name="OutputParameterRef" type="tc:ParameterRefType"/>
        <element name="Trigger" type="tc:TriggerType"/>
      </sequence>
      <attribute name="trigger" type="tc:NameReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>

<complexType name="ReedSolomonType">
  <annotation>
    <documentation>Abstract type used by both ReedSolomonEncoder and ReedSolomonDecoder</documentation>
  </annotation>
  <sequence>
    <element name="InputParameterRef" type="tc:ParameterRefType"/>
    <element name="OutputParameterRef" type="tc:ParameterRefType"/>
    <element name="GeneratorPolynomial" type="tc:PolynomialType"/>
  </sequence>
  <attribute name="inputParameterRef" type="tc:NameReferenceType" use="required"/>
  <attribute name="outputParameterRef" type="tc:NameReferenceType" use="required"/>
</complexType>

<complexType name="TriggerType">
  <annotation>
    <documentation>A trigger is used to initiate the processing of some algorithm. A trigger may be based on an update of a Parameter or on a time basis. Triggers may also have a rate that limits their firing to a 1/rate basis.</documentation>
  </annotation>
  <choice maxOccurs="unbounded">
    <element name="ParameterRef" minOccurs="0">
      <annotation>
        <documentation>Names a parameter that will start the execution of the algorithm. There are two attributes to TriggerParm, triggerName and parameterName. parameterName is a parameter reference name for a parameter that when it changes, will cause this algorithm to be executed. triggerName is an optional "friendly" name for the trigger.</documentation>
      </annotation>
    </element>
  </choice>
</complexType>
<extension base="tc:ParameterRefType"/>
</complexType>
</element>
</choice>
<attribute name="name" type="string" use="optional"/>
<attribute name="triggerRate" type="nonNegativeInteger" use="optional" default="1"/>
</complexType>
</schema>
StreamDefinitions.xsd

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Nori Jungmann (Engineering) -->
  <!-- unqualified element and attribute names -->
  <annotation>
    <documentation>$Id: StreamDefinitions.xsd,v 1.11 2003/03/03 14:34:40 gerry Exp $</documentation>
    <documentation xml:lang="en">
      This schema provides a language for defining binary stream data.
    </documentation>
  </annotation>
  <include schemaLocation="CommonTypes.xsd"/>
  <simpleType name="StreamSourceType">
    <restriction base="string">
      <enumeration value="TC"/>
      <enumeration value="TM"/>
    </restriction>
  </simpleType>
  <complexType name="PCMStreamType">
    <annotation>
      <documentation xml:lang="en">A PCM Stream Type is the high level definition for all Pulse Code Modulated (PCM) (i.e., binary) streams.</documentation>
    </annotation>
    <sequence>
      <element name="LongDescription" type="tc:LongDescriptionType" minOccurs="0"/>
    </sequence>
    <attribute name="name" type="tc:NameType"/>
    <attribute name="shortDescription" type="tc:ShortDescriptionType"/>
    <attribute name="bitRateInBPS" type="double"/>
    <attribute name="pcmType" default="NRZL">
      <simpleType>
        <restriction base="string">
          <enumeration value="NRZL"/>
          <enumeration value="NRZM"/>
          <enumeration value="NRZS"/>
          <enumeration value="BiPhaseL"/>
          <enumeration value="BiPhaseM"/>
          <enumeration value="BiPhaseS"/>
        </restriction>
      </simpleType>
    </attribute>
    <attribute name="inverted" type="boolean" default="false"/>
  </complexType>
  <complexType name="FixedFrameStreamType">
    <annotation>
      <documentation xml:lang="en">The top level type definition for all data streams with a fixed frame length.</documentation>
    </annotation>
    <complexContent>
      <extension base="tc:PCMStreamType">
        <sequence>
          <element name="SyncStrategy" type="tc:SyncStrategyType"/>
        </sequence>
        <attribute name="frameLengthInBits" type="long" use="required"/>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="VariableFrameStreamType">
    <annotation>
      <documentation xml:lang="en">The top level type definition for all data streams with a variable frame length.</documentation>
    </annotation>
    <complexContent>
      <extension base="tc:PCMStreamType">
        <sequence>
          <element name="SyncStrategy" type="tc:SyncStrategyType"/>
        </sequence>
        <attribute name="zeroBitInsertion" type="boolean" use="optional" default="true"/>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="EncryptedStreamType">
    <annotation>
      <documentation xml:lang="en">Encrypted stream type definition.</documentation>
    </annotation>
  </complexType>
</schema>
<documentation xml:lang="en">The top level type definition for all encrypted streams.</documentation>
</annotation>
<complexContent>
<extension base="tc:PCMStreamType">
<attribute name="algorithmName" type="string" use="required"/>
<attribute name="key" type="string"/>
</extension>
</complexContent>
</complexType>
<complexType name="ConvolutionalStreamType">
<annotation>
<documentation xml:lang="en">The type definition for convolutionally encoded streams. The processing algorithm for the convolutional encoder is described using a series of generator polynomials. Each polynomial, corresponds to a modulo two adder and the order of the polynomials corresponds to the order in the output stage. Each term in each polynomial corresponds to a register in the shift register. The coefficients in these polynomials are either "1" for connected or "0" for not connected.</documentation>
</annotation>
<complexContent>
<extension base="tc:PCMStreamType">
<sequence>
<element name="Polynomial" type="tc:PolynomialType" minOccurs="2" maxOccurs="unbounded"/>
</sequence>
</extension>
</complexContent>
</complexType>
<complexType name="StreamType">
<annotation>
<documentation xml:lang="en">A container Type used to contain a single reference to all known Stream types</documentation>
</annotation>
<choice maxOccurs="unbounded">
<element name="FixedFrameStream" type="tc:FixedFrameStreamType"/>
<element name="EncryptedStream" type="tc:EncryptedStreamType"/>
<element name="VariableFrameStream" type="tc:VariableFrameStreamType"/>
<element name="ConvolutionalStream" type="tc:ConvolutionalStreamType"/>
<element name="NonStandardData" type="tc:NonStandardDataType"/>
</choice>
<attribute name="streamSource" type="tc:StreamSourceType" use="required"/>
</complexType>
<complexType name="SyncStrategyType">
<annotation>
<documentation>A Sync Strategy specifies the requirements to deem a PCM Fixed Frame Stream "in-sync" or out of sync.</documentation>
</annotation>
<sequence>
<element name="AutoInvert" minOccurs="0">
<annotation>
<documentation xml:lang="en">After a certain number of bad frames, it may be desirable to invert the incoming data, and then look for frame sync.</documentation>
</annotation>
<complexType>
<attribute name="badFramesToAutoInvert" type="positiveInteger" default="2"/>
<attribute name="algorithmName" type="string"/>
</complexType>
</element>
<element name="MessageMatch" minOccurs="0" maxOccurs="unbounded">
<complexType>
<sequence>
<element name="NameRef" type="tc:NameReferenceType" maxOccurs="unbounded">
<annotation>
<documentation xml:lang="en">Reference to an existing message</documentation>
</annotation>
</element>
</sequence>
<attribute name="bitLocation" type="integer" default="0"/>
</complexType>
</element>
<element name="SyncPattern">
<annotation>
<documentation xml:lang="en">The pattern of bits used to look for frame synchronization.</documentation>
</annotation>
<complexType>
<attribute name="pattern" type="hexBinary" use="required"/>
</complexType>
</element>
</sequence>
</complexType>
<complexType>
  <attribute name="bitLocation" type="integer" default="0"/>
  <attribute name="maskLengthInBits" type="positiveInteger"/>
  <attribute name="patternLengthInBits" type="positiveInteger" use="required"/>
</complexType>
</element>
</sequence>
<attribute name="verifyToLockGoodFrames" type="integer" default="4"/>
<attribute name="checkToLockGoodFrames" type="integer" default="1"/>
<attribute name="maxErrorsInSyncWord" type="integer" default="0"/>
<attribute name="syncAperature" type="integer" default="0"/>
</complexType>
</schema>
  <annotation>
    <documentation>$Id: Packaging.xsd,v 1.17 2003/01/24 11:42:43 cooper Exp $</documentation>
    <documentation xml:lang="en">This schema defines the dictionary for messages and containers, which in turn describe the physical composition of data in a communication system</documentation>
  </annotation>
  <include schemaLocation="Parameter.xsd"/>
  <simpleType name="ServiceNameType">
    <annotation>
      <documentation xml:lang="en">Identifier for a service</documentation>
    </annotation>
    <restriction base="tc:NameType"/>
  </simpleType>
  <simpleType name="MessageKeyIdType">
    <annotation>
      <documentation xml:lang="en">Identifier for a message key</documentation>
    </annotation>
    <restriction base="tc:NameType"/>
  </simpleType>
  <simpleType name="MessageNameType">
    <annotation>
      <documentation xml:lang="en">Identifier for a message</documentation>
    </annotation>
    <restriction base="tc:NameType"/>
  </simpleType>
  <simpleType name="ContainerNameType">
    <annotation>
      <documentation xml:lang="en">Identifier for a container</documentation>
    </annotation>
    <restriction base="tc:NameType"/>
  </simpleType>
  <complexType name="MessageKeyType">
    <annotation>
      <documentation xml:lang="en">Denotes a bit field in a message that can be used to identify a particular message</documentation>
    </annotation>
    <sequence>
      <element name="LongDescription" type="tc:LongDescriptionType" minOccurs="0">
        <annotation>
          <documentation xml:lang="en">Optional long key description</documentation>
        </annotation>
      </element>
      <element name="OffsetInBits" type="tc:BitOffsetType" minOccurs="0">
        <annotation>
          <documentation xml:lang="en">Offset relative to start of container representing this message</documentation>
        </annotation>
      </element>
    </sequence>
    <attribute name="Id" type="tc:MessageKeyIdType" use="required">
      <annotation>
        <documentation xml:lang="en">Id of message key field</documentation>
      </annotation>
    </attribute>
    <attribute name="ShortDescription" type="tc:ShortDescriptionType" use="optional">
      <annotation>
        <documentation xml:lang="en">Optional short key description</documentation>
      </annotation>
    </attribute>
    <attribute name="WidthInBits" type="positiveInteger" use="required">
      <annotation>
        <documentation xml:lang="en">Width of key field in bits</documentation>
      </annotation>
    </attribute>
  </complexType>
</schema>
<element name="LongDescription" type="tc:LongDescriptionType" minOccurs="0">
  <documentation xml:lang="en">Optional long description of container</documentation>
</element>

<element name="ImplementsMessage" type="tc:MessageKeyMatchListType" minOccurs="0">
  <documentation xml:lang="en">Optional message this container represents</documentation>
</element>

<element name="Sequence" type="tc:SequenceType" minOccurs="0">
  <documentation xml:lang="en">Optional body of this container. Most container will have a body however event packets do not.</documentation>
</element>

<attribute name="name" type="tc:ContainerNameType" use="required">
  <documentation xml:lang="en">Id of container</documentation>
</attribute>

<attribute name="shortDescription" type="tc:ShortDescriptionType" use="optional">
  <documentation xml:lang="en">Optional short description of container</documentation>
</attribute>

<complexType name="ContainerRefType">
  <documentation xml:lang="en">Holds an offset to a referred to container and an optional parameter to determine validity of this container</documentation>
  <sequence>
    <element name="Occurs" type="tc:OccursType" minOccurs="0" maxOccurs="unbounded">
      <documentation xml:lang="en">Repeating group to include</documentation>
    </element>
    <element name="DependantOn" type="tc:ComparisonCheckType" minOccurs="0">
      <documentation xml:lang="en">Parameter that the inclusion of the sub-container depends on</documentation>
    </element>
    <element name="OffsetInBits" type="tc:BitOffsetType" minOccurs="0">
      <documentation xml:lang="en">Offset, in bits, from the start of this container to start sub-container. If not present then sub-container start either at offset 0 if the first sub-container or relative to end of previous object</documentation>
    </element>
  </sequence>
  <attribute name="nameRef" type="tc:ContainerNameType" use="required">
    <documentation xml:lang="en">Id of container to contain</documentation>
  </attribute>
</complexType>

<complexType name="DynamicContainerRefType">
  <documentation xml:lang="en">Holds a reference to a parameter that will hold a container id at runtime to expand</documentation>
  <sequence>
    <element name="Occurs" type="tc:OccursType" minOccurs="0" maxOccurs="unbounded">
      <documentation xml:lang="en">Repeating group to include</documentation>
    </element>
    <element name="DependantOn" type="tc:ComparisonCheckType" minOccurs="0">
      <documentation xml:lang="en">Parameter that the inclusion of the sub-container depends on</documentation>
    </element>
  </sequence>
  <attribute name="nameRef" type="tc:ContainerNameType" use="required">
    <documentation xml:lang="en">Id of container to contain</documentation>
  </attribute>
</complexType>
<complexType>
    <sequence>
        <element name="Message" type="tc:MessageType" maxOccurs="unbounded">
            <annotation>
                <documentation xml:lang="en">A single message definition</documentation>
            </annotation>
        </element>
    </sequence>
</complexType>

<key name="MessageKey">
    <selector xpath="Message"/>
    <field xpath="Id"/>
</key>

<element name="ContainerList" minOccurs="0">
    <annotation>
        <documentation xml:lang="en">Holds the list of container definitions</documentation>
    </annotation>
    <complexType>
        <sequence>
            <element name="Container" type="tc:ContainerType" maxOccurs="unbounded">
                <annotation>
                    <documentation xml:lang="en">A single container definition</documentation>
                </annotation>
            </element>
        </sequence>
    </complexType>
    <key name="ContainerKey">
        <selector xpath="Container"/>
        <field xpath="Id"/>
    </key>
</element>

<element name="NonStandardData" type="tc:NonStandardDataType" minOccurs="0"/>

<complexType name="SequenceType">
    <annotation>
        <documentation>Used to describe a sequence of sequences, containers, parameters, or choices</documentation>
    </annotation>
    <sequence>
        <element name="Occurs" type="tc:OccursType" minOccurs="0" maxOccurs="unbounded">
            <annotation>
                <documentation xml:lang="en">Repeating group to include</documentation>
            </annotation>
        </element>
        <choice maxOccurs="unbounded">
            <element name="ContainerRef" type="tc:ContainerRefType"/>
            <element name="DynamicContainerRef" type="tc:DynamicContainerRefType"/>
            <element name="ArgumentRef" type="tc:OffsetParameterRefType">
                <annotation>
                    <documentation xml:lang="en">Set of command arguments to include</documentation>
                </annotation>
            </element>
            <element name="ParameterRef" type="tc:OffsetParameterRefType">
                <annotation>
                    <documentation xml:lang="en">Set of parameters to include</documentation>
                </annotation>
            </element>
            <element name="DynamicParameterRef" type="tc:OffsetParameterRefType">
                <annotation>
                    <documentation xml:lang="en">Holds a reference to a parameter that will hold an onboard Id</documentation>
                </annotation>
            </element>
            <element name="BinaryConstant">
                <annotation>
                    <documentation>May also use a reference to a constant parameter</documentation>
                </annotation>
            </element>
        </choice>
    </sequence>
</complexType>
<sequence>
  <element name="LongDescription" type="tc:LongDescriptionType" minOccurs="0">
    <annotation>
      <documentation xml:lang="en">Optional long description of fixed area</documentation>
    </annotation>
  </element>
  <element name="OffsetInBits" type="tc:BitOffsetType" minOccurs="0">
    <attribute name="lengthInBits" type="nonNegativeInteger" use="required"/>
    <attribute name="value" type="hexBinary" use="required"/>
  </element>
</sequence>

<complexType name="ChoiceType">
  <annotation>
    <documentation>Used to one or more possible container choices.</documentation>
  </annotation>
  <sequence>
    <element name="Occurs" type="tc:OccursType" minOccurs="0" maxOccurs="unbounded">
      <annotation>
        <documentation xml:lang="en">Repeating group to include</documentation>
      </annotation>
    </element>
    <element name="ContainerRef" type="tc:ContainerRefType" maxOccurs="unbounded"/>
  </sequence>
</complexType>
</schema>
  <annotation>
    <documentation>$Id: CommonTypes.xsd,v 1.16 2003/03/03 07:51:22 cooper Exp $</documentation>
    <documentation xml:lang="en">Schema for global type definitions used in the Space TnC specification</documentation>
  </annotation>
  <include schemaLocation="Algorithm.xsd"/>
  <!-- Basic elements used for in all dictionaries -->
  <complexType name="AbsoluteTimeType">
    <annotation>
      <documentation>Contains an absolute (to a known epoch) time</documentation>
    </annotation>
    <simpleContent>
      <extension base="dateTime"/>
    </simpleContent>
  </complexType>
  <complexType name="AliasType">
    <annotation>
      <documentation xml:lang="en">Used to save an alias (alternate) name or ID that may be used to identify the parameter by the factory or in an alternative ground system. Some ground system processing equipent has some severe naming restrictions on parameters (e.g., names must less then 12 characters, single case or integral id's only). </documentation>
    </annotation>
    <attribute name="nameSet" type="string" use="required"/>
    <attribute name="alias" type="string" use="required"/>
  </complexType>
  <complexType name="BaseDataType">
    <annotation>
      <documentation>Used to describe the base data types used by the ground system. The ValidRange is a range for the parameter's value that is valid. It is used to display graphs and/or gauges that have pre-defined boundaries</documentation>
    </annotation>
    <sequence>
      <element name="Any">
        <annotation>
          <documentation>A place holder parameter that is populated at runtime by using the value currently contained in the SourceParameter as an onboard Id</documentation>
        </annotation>
      </complexType>
      <complexType>
        <sequence>
          <element name="SourceParameterRef" type="tc:NameReferenceType"/>
        </sequence>
      </complexType>
    </element>
    <element name="Float">
      <complexType>
        <sequence>
          <element name="DefaultValue" type="tc:DefaultValueType" minOccurs="0"/>
          <element name="ValidRange" type="tc:RangeType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="Integer">
      <complexType>
        <sequence>
          <element name="DefaultValue" type="tc:DefaultValueType" minOccurs="0"/>
          <element name="ValidRange" type="tc:RangeType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="Enumerated">
      <complexType>
        <sequence>
          <element name="DefaultValue" type="tc:DefaultValueType" minOccurs="0"/>
          <element name="ValidRange" type="tc:RangeType" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </complexType>
    </element>
    <element name="NameReferenceType">
      <complexType>
        <sequence>
          <element name="SourceParameterRef" type="tc:NameReferenceType"/>
        </sequence>
      </complexType>
    </element>
  </sequence>
</documentation>
</complexType>
<complexType>
  <element name="Binary">
    <complexType>
      <sequence>
        <element name="DefaultValue" type="tc:DefaultValueType" minOccurs="0"/>
        <element name="ValidRange" type="tc:RangeType" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="lengthInBits" type="nonNegativeInteger"/>
    </complexType>
  </element>

  <element name="Boolean">
    <complexType>
      <sequence>
        <element name="DefaultValue" type="tc:DefaultValueType" minOccurs="0"/>
        <element name="ValidRange" type="tc:RangeType" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>

  <element name="String">
    <complexType>
      <sequence>
        <element name="DefaultValue" type="tc:DefaultValueType" minOccurs="0"/>
        <element name="Enumeration" type="string" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="maxLength" type="nonNegativeInteger" use="optional"/>
    </complexType>
  </element>

  <element name="AbsoluteTime">
    <complexType>
      <sequence>
        <element name="DefaultValue" type="tc:DefaultValueType" minOccurs="0"/>
      </sequence>
      <annotation>
        <documentation>Use the [ISO 8601] extended format CCYy-MM-DDThh:mm:ss where "CC" represents the century, "YY" the year, "MM" the month and "DD" the day, preceded by an optional leading '-' sign to indicate a negative number. If the sign is omitted, "+" is assumed. The letter "T" is the date/time separator and "hh", "mm", "ss" represent hour, minute and second respectively. Additional digits can be used to increase the precision of fractional seconds if desired i.e the format ss.ss... with any number of digits after the decimal point is supported.</documentation>
      </annotation>
    </complexType>
  </element>

  <element name="RelativeTime">
    <complexType>
      <sequence>
        <element name="DefaultValue" type="tc:DefaultValueType" minOccurs="0"/>
      </sequence>
      <annotation>
        <documentation>Duration's are expressed as PnYn MnDTnHnMnS, where nY represents the number of years, nM the number of months, nD the number of days, 'T' is the date/time separator, nH the number of hours, nM the number of minutes and nS the number of seconds. The number of seconds can include decimal digits to arbitrary precision. For example, to indicate a duration of 1 year, 2 months, 3 days, 10 hours, and 30 minutes, one would write: P1Y2M3DT10H30M. One could also indicate a duration of minus 120 days as: -P120D.</documentation>
      </annotation>
    </complexType>
  </element>

  <element name="Units">
    <complexType>
      <sequence>
        <element name="Unit" type="tc:UnitType" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>

  <element name="Unitless"/>
</choice>
</choice>
</complexType>
<complexType name="BaseParameterType">
  <annotation>
    <documentation>A type definition used as the base type for a CommandDefinition and for BusCommandDefinition.</documentation>
  </annotation>
  <complexContent>
    <extension base="tc:BaseDataType">
      <sequence>
        <element name="LongDescription" type="tc:LongDescriptionType" minOccurs="0">
          <annotation>
            <documentation>An arbitrarily long description for the Parameter. May also include HTML markup.</documentation>
          </annotation>
        </element>
        <element name="BusAttributes" type="tc:BusAttributesType" minOccurs="0"/>
      </sequence>
      <attribute name="name" type="tc:NameType" use="required"/>
      <attribute name="shortDescription" type="tc:ShortDescriptionType" use="optional"/>
    </extension>
  </complexContent>
</complexType>

<complexType name="DefaultValueType" mixed="true">
  <annotation>
    <documentation xml:lang="en">Holds a default value and indicated whether it is calibrated or not</documentation>
  </annotation>
  <attribute name="calibrated" type="boolean" default="false"/>
</complexType>

<complexType name="BitOffsetType">
  <annotation>
    <documentation xml:lang="en">Denotes a bit field in a message that can be used to identify a particular message</documentation>
  </annotation>
  <choice>
    <element name="AbsoluteOffset" type="nonNegativeInteger" default="0" nillable="true">
      <annotation>
        <documentation xml:lang="en">Absolute offset in bits. Default is 0.</documentation>
      </annotation>
    </element>
    <element name="RelativeOffset" type="integer" default="0" nillable="true">
      <annotation>
        <documentation xml:lang="en">Relative offset in bits from end of previous object, can be negative. Default is 0</documentation>
      </annotation>
    </element>
  </choice>
</complexType>

<complexType name="BusAttributesType">
  <annotation>
    <documentation xml:lang="en">Bus attributes describe how a particular piece of data is sent or received from some non-native, off-platform device. (e.g. a spacecraft)</documentation>
  </annotation>
  <sequence>
    <element name="ErrorDetectCorrect" type="tc:ErrorDetectCorrectType" minOccurs="0"/>
    <element name="DefaultCalibrator" type="tc:CalibratorType" minOccurs="0"/>
    <element name="ByteOrder" minOccurs="0">
      <annotation>
        <documentation>Used to describe an arbitrary byte order in multibyte parameters. order is the order the byte is transmitted.</documentation>
      </annotation>
      <complexType>
        <sequence maxOccurs="unbounded">
          <element name="Byte">
            <complexType>
              <attribute name="appearanceInStream" type="nonNegativeInteger" use="required"/>
              <attribute name="byteSignificance" type="nonNegativeInteger" use="required"/>
            </complexType>
          </element>
        </sequence>
      </complexType>
    </element>
    <element name="ContextCalibrator" minOccurs="0" maxOccurs="unbounded">
      <annotation>
        <documentation>Use when different calibrations must be used on the Parameter in different contexts. Use the first one that tests true</documentation>
      </annotation>
    </element>
  </sequence>
</complexType>
<sequence>
  <element name="Context" type="tc:ComparisonCheckType"/>
  <element name="Calibrator" type="tc:CalibratorType"/>
</sequence>
</complexType>
</element>
</sequence>
</attribute>
<attribute name="bitOrder" default="MostSignificantBitFirst">
  <simpleType>
    <restriction base="string">
      <enumeration value="MostSignificantBitFirst"/>
      <enumeration value="LeastSignificantBitFirst"/>
    </restriction>
  </simpleType>
</attribute>
<attribute name="busType" use="required">
  <simpleType>
    <restriction base="string">
      <enumeration value="Binary"/>
      <enumeration value="OnesComp"/>
      <enumeration value="TwosComp"/>
      <enumeration value="SignMag"/>
      <enumeration value="Unsigned"/>
      <enumeration value="IEEE754_1985Float"/>
      <enumeration value="MILSTD_1750AFloat"/>
      <enumeration value="ASCII_String"/>
    </restriction>
  </simpleType>
</attribute>
</complexType>
<complexType name="SimpleParameterRefType">
  <annotation>
    <documentation xml:lang="en">A simple reference to a Parameter</documentation>
  </annotation>
  <attribute name="parameterName" type="tc:NameReferenceType" use="required"/>
  <attribute name="spaceSystemName" type="string"/>
</complexType>
<complexType name="ComparisonCheckType">
  <annotation>
    <documentation xml:lang="en">A simple boolean expression</documentation>
  </annotation>
  <sequence>
    <element name="ParameterRef" type="tc:SimpleParameterRefType"/>
    <element name="ComparisonOperator" type="tc:ComparisonOperatorsType"/>
    <choice>
      <element name="ArgumentRef" type="tc:SimpleParameterRefType">
        <annotation>
          <documentation>Takes the value of a command argument, is assumed to be of the same type as the comparison Parameter</documentation>
        </annotation>
      </element>
      <element name="ParameterRef" type="tc:SimpleParameterRefType">
        <annotation>
          <documentation>Parameter is assumed to be of the same type as the comparison Parameter</documentation>
        </annotation>
      </element>
      <element name="Value" type="string">
        <annotation>
          <documentation>Value is assumed to be of the same type as the comparison Parameter</documentation>
        </annotation>
      </element>
    </choice>
  </sequence>
  <attribute name="calibratedComparison" type="boolean" default="false">
    <annotation>
      <documentation>Should the compaison be based on the calibrated format or not, if true and a constant value is used this should be in calibrated format too.</documentation>
    </annotation>
  </attribute>
</complexType>
Operators to use when testing a boolean condition for a validity check.

<documentation xml:lang="en">A simple element that provides for simple, but common error checking and detection.</documentation>

<complexType name="ErrorDetectCorrectType">
  <annotation>
    <documentation xml:lang="en">A simple element that provides for simple, but common error checking and detection.</documentation>
  </annotation>
  <choice>
    <element name="Parity">
      <annotation>
        <documentation xml:lang="en">Bit position starts with 'zero'.</documentation>
      </annotation>
      <complexType>
        <attribute name="type" use="required">
          <simpleType>
            <restriction base="string">
              <enumeration value="Even"/>
              <enumeration value="Odd"/>
            </restriction>
          </simpleType>
        </attribute>
        <attribute name="bitsFromStart" type="nonNegativeInteger" use="required"/>
      </complexType>
    </element>
    <element name="CRC">
      <annotation>
        <documentation xml:lang="en">Cyclic Redundancy Check definition. Legal values for coefficient's are 0 or 1. Exponents must be integer values.</documentation>
      </annotation>
      <complexType>
        <sequence>
          <element name="PolynomialType" type="tc:PolynomialType"/>
        </sequence>
        <attribute name="bitsFromStart" type="nonNegativeInteger"/>
      </complexType>
    </element>
  </choice>
</complexType>

<complexType name="FormatType">
  <annotation>
    <documentation xml:lang="en">Add</documentation>
  </annotation>
  <restriction base="string">
    <enumeration value="SignedInteger"/>
    <enumeration value="UnsignedInteger"/>
    <enumeration value="Real"/>
  </restriction>
</complexType>

<complexType name="HeaderType">
  <annotation>
    <documentation xml:lang="en">Schema for a Header record. A header contains general information about the document or subdocument.</documentation>
  </annotation>
  <sequence>
    <element name="Author" type="string" minOccurs="0" maxOccurs="unbounded"/>
    <element name="Note" type="string" minOccurs="0" maxOccurs="unbounded"/>
    <element name="History" type="string" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="version" type="string"/>
  <attribute name="date" type="string"/>
  <attribute name="classification" default="Unclassified"/>
</complexType>
<simpleType name="NameType">
  
  <documentation xml:lang="en">Used for "directory" style unique names. Only letters, digits, ",", ",", and "," are allowed</documentation>
  
  <restriction base="string">
    <pattern value="^[a-zA-Z0-9-\_ ]*$"/>
  </restriction>
</simpleType>

<simpleType name="NameReferenceType">
  
  <documentation xml:lang="en">Used when referencing a directory style "NameType".</documentation>
  
  <restriction base="string">
    <pattern value="^[a-zA-Z0-9-\_ ]*$"/>
  </restriction>
</simpleType>

<complexType name="NonStandardDataType">
  
  <annotation>
    <documentation>An any type that may be used in certain key locations within the database to hold mission unique data.</documentation>
  </annotation>
  
  <complexContent>
    <extension base="anyType">
      <attribute name="usage" type="string" use="required"/>
    </extension>
  </complexContent>
</complexType>

<complexType name="PolynomialType">
  
  <annotation>
    <documentation xml:lang="en">A polynomial expression. For example: 3 + 2x</documentation>
  </annotation>
  
  <sequence>
    <element name="Term" maxOccurs="unbounded">
      
      <annotation>
        <documentation xml:lang="en">A term in a polynomial expression.</documentation>
      </annotation>
      
      <complexType>
        <attribute name="coefficient" type="double" use="required"/>
        <attribute name="exponent" type="double" use="required"/>
      </complexType>
    </element>
  </sequence>
</complexType>

<complexType name="PropertyType">
  
  <annotation>
    <documentation>Used for custom user properties</documentation>
  </annotation>
  
  <sequence minOccurs="0" maxOccurs="unbounded">
    <element name="Property" type="tc:PropertyType"/>
  </sequence>
  
  <attribute name="name" type="tc:NameReferenceType" use="required"/>
  
  <attribute name="shortDescription" type="string"/>
  
  <attribute name="longDescription" type="string"/>
  
  <attribute name="value" type="string" use="required"/>
</complexType>

<simpleType name="RadixType">
  
  <annotation>
    <documentation xml:lang="en">Specifies the number base</documentation>
  </annotation>
  
  <restriction base="string">
    <enumeration value="Decimal"/>
    <enumeration value="Hexadecimal"/>
    <enumeration value="Octal"/>
    <enumeration value="Binary"/>
  </restriction>
</simpleType>

<complexType name="RangeType">
  
  <annotation>

  </annotation>
</complexType>
A range of numbers. "minInclusive", "minExclusive", "maxInclusive" and "maxExclusive" attributes are borrowed from the W3C schema language.

```
<complexType>
  <documentation xml:lang="en">A range of numbers. "minInclusive", "minExclusive", "maxInclusive" and "maxExclusive" attributes are borrowed from the W3C schema language.</documentation>
  <attribute name="minInclusive" type="string"/>
  <attribute name="minExclusive" type="string"/>
  <attribute name="maxInclusive" type="string"/>
  <attribute name="maxExclusive" type="string"/>
  <attribute name="calibrated" type="boolean" default="false"/>
</complexType>
```

```
<complexType name="ShortDescriptionType">
  <annotation>
    <documentation xml:lang="en">A bounded description</documentation>
  </annotation>
  <restriction base="string">
    <maxLength value="32"/>
  </restriction>
</complexType>
```

```
<complexType name="SplinePointType">
  <annotation>
    <documentation xml:lang="en">A spline is a set of points from which a curve may be drawn to interpolate raw to calibrated values.</documentation>
  </annotation>
  <attribute name="order" type="positiveInteger" default="1"/>
  <attribute name="raw" type="double" use="required"/>
  <attribute name="calibrated" type="double" use="required"/>
</complexType>
```

```
<complexType name="SystemNameType">
  <annotation>
    <documentation xml:lang="en">Sub-system name.</documentation>
  </annotation>
  <simpleContent>
    <extension base="string"/>
  </simpleContent>
</complexType>
```

```
<complexType name="ToStringType">
  <annotation>
    <documentation xml:lang="en">There are two ways numeric data can be changed to string data: using a C style format string, or using an enumerated list. Enumerated lists can be assigned to a single value or a value range.</documentation>
  </annotation>
  <choice>
    <choice maxOccurs="unbounded">
      <element name="ValueEnumeration">
        <complexType>
          <attribute name="value" type="integer" use="required"/>
          <attribute name="stringValue" type="string" use="required"/>
        </complexType>
      </element>
      <element name="RangeEnumeration">
        <complexType>
          <sequence>
            <element name="Range" type="tc:RangeType"/>
          </sequence>
          <attribute name="stringValue" type="string" use="required"/>
        </complexType>
      </element>
    </choice>
    <element name="NumberFormat">
      <complexType>
        <attribute name="numberBase" type="tc:RadixType" use="optional"/>
        <attribute name="minimumFractionDigits" type="nonNegativeInteger" use="optional"/>
        <attribute name="maximumFractionDigits" type="nonNegativeInteger" use="optional"/>
        <attribute name="minimumIntegerDigits" type="nonNegativeInteger" use="optional"/>
        <attribute name="maximumIntegerDigits" type="nonNegativeInteger" use="optional"/>
        <attribute name="negativeSuffix" type="string" use="optional"/>
        <attribute name="positiveSuffix" type="string" use="optional"/>
        <attribute name="negativePrefix" type="string" use="optional" default="-"/>
        <attribute name="positivePrefix" type="string" use="optional"/>
      </complexType>
    </element>
  </choice>
</complexType>
```
<attribute name="showThousandsGrouping" type="boolean" use="optional" default="true"/>
<attribute name="notation" use="optional" default="normal">
	<simpleType>
		<restriction base="string">
			<enumeration value="normal"/>
			<enumeration value="scientific"/>
			<enumeration value="engineering"/>
		</restriction>
	</simpleType>
</attribute>
</complexType>

<complexType name="UnitType" mixed="true">
	<annotation>
		<documentation>Used to hold the unit(s) plus possibly the exponents for the units</documentation>
	</annotation>
	<attribute name="power" type="decimal" use="optional" default="1"/>
	<attribute name="description" type="string"/>
</complexType>

<complexType name="RelativeTimeType">
	<annotation>
		<documentation>Used to describe a relative time. Normally used for time offsets. A Relative time is expressed as PnYn MnDTnH nMnS, where nY represents the number of years, nM the number of months, nD the number of days, 'T' is the date/time separator, nH the number of hours, nM the number of minutes and nS the number of seconds. The number of seconds can include decimal digits to arbitrary precision. For example, to indicate a duration of 1 year, 2 months, 3 days, 10 hours, and 30 minutes, one would write: P1Y2M3DT10H30M. One could also indicate a duration of minus 120 days as: -P120D.</documentation>
	</annotation>
	<restriction base="duration"/>
</complexType>

<complexType name="RealRangeType">
	<annotation>
		<documentation>A range for real numbers. "minInclusive", "minExclusive", "maxInclusive" and "maxExclusive" attributes are borrowed from the W3C schema language.</documentation>
	</annotation>
	<attribute name="minInclusive" type="float"/>
	<attribute name="minExclusive" type="float"/>
	<attribute name="maxInclusive" type="float"/>
	<attribute name="maxExclusive" type="float"/>
</complexType>

<complexType name="OccursType">
	<annotation>
		<documentation xml:lang="en">Hold a structure that can be repeated X times, where X is held in the supplied parameter reference</documentation>
	</annotation>
	<sequence>
		<choice>
			<element name="CountRef" type="tc:OffsetParameterRefType">
			<annotation>
				<documentation xml:lang="en">Parameter that contains the count of repeated structures, implies variable length structure</documentation>
			</annotation>
			</element>
			<element name="Count" type="integer" use="optional" default="0">
			<annotation>
				<documentation xml:lang="en">Fixed value that contains the count of repeated structures, implies fixed length structure</documentation>
			</annotation>
		</choice>
		<attribute name="NumOFOcc" type="positiveInteger" use="required">
		<annotation>
			<documentation xml:lang="en">Number of occurrences of structure in container</documentation>
		</annotation>
	</attribute>
	<attribute name="RelativeOffsetInBits" type="nonNegativeInteger" use="optional">
	<annotation>
		<documentation xml:lang="en">Number of bits between the start of two consecutive structures</documentation>
	</annotation>
	</attribute>
	<attribute name="TimeOffset" type="integer" use="optional" default="0">
	<annotation>
	</attribute>
</complexType>
<documentation xml:lang="en">Time delay in milliseconds between two consecutive occurrences</documentation>
</annotation>
</complexType>
</element>
</complexType>
<complexType name="IntegerRangeType">
<annotation>
<documentation>A range for integers</documentation>
</annotation>
<attribute name="min" type="long"/>
<attribute name="max" type="long"/>
</complexType>
</schema>