Tutorial on SDTF Specifications

Four-letter acronyms that make a difference in ground system interoperability

GSAW MMXVI

Prepared By The OMG Space Domain Task Force
And

AMERGINT TECHNOLOGIES

KRATOS

HARRIS

Published by The Aerospace Corporation with permission.

OMG OBJECT MANAGEMENT GROUP

Tutorial Welcome and Introductions

Gerry Simon
Kratos Defense Systems
Architect
Former SDTF chair - XTCE

Brad Kizzort
Harris Corporation Space
Systems Engineer
Current SDTF co-chair – SOLM & XUSP

Rob Andzik
Amergint Technologies Inc.
President
Current SDTF co-chair – GEMS

Specs We Will Cover:

- XML Telemetric and Command Exchange (XTCE)
- XTCE US government satellite conformance Profile (XUSP)
- Ground Equipment Monitoring Service (GEMS)
- Satellite Operations Language Metamodel (SOLM)

What Are Your Expectations for the Tutorial?

XTCE Tutorial

Four-letter acronyms that make a difference in ground system interoperability

GSAW MMXVI

Prepared By The OMG Space Domain Task Force

And

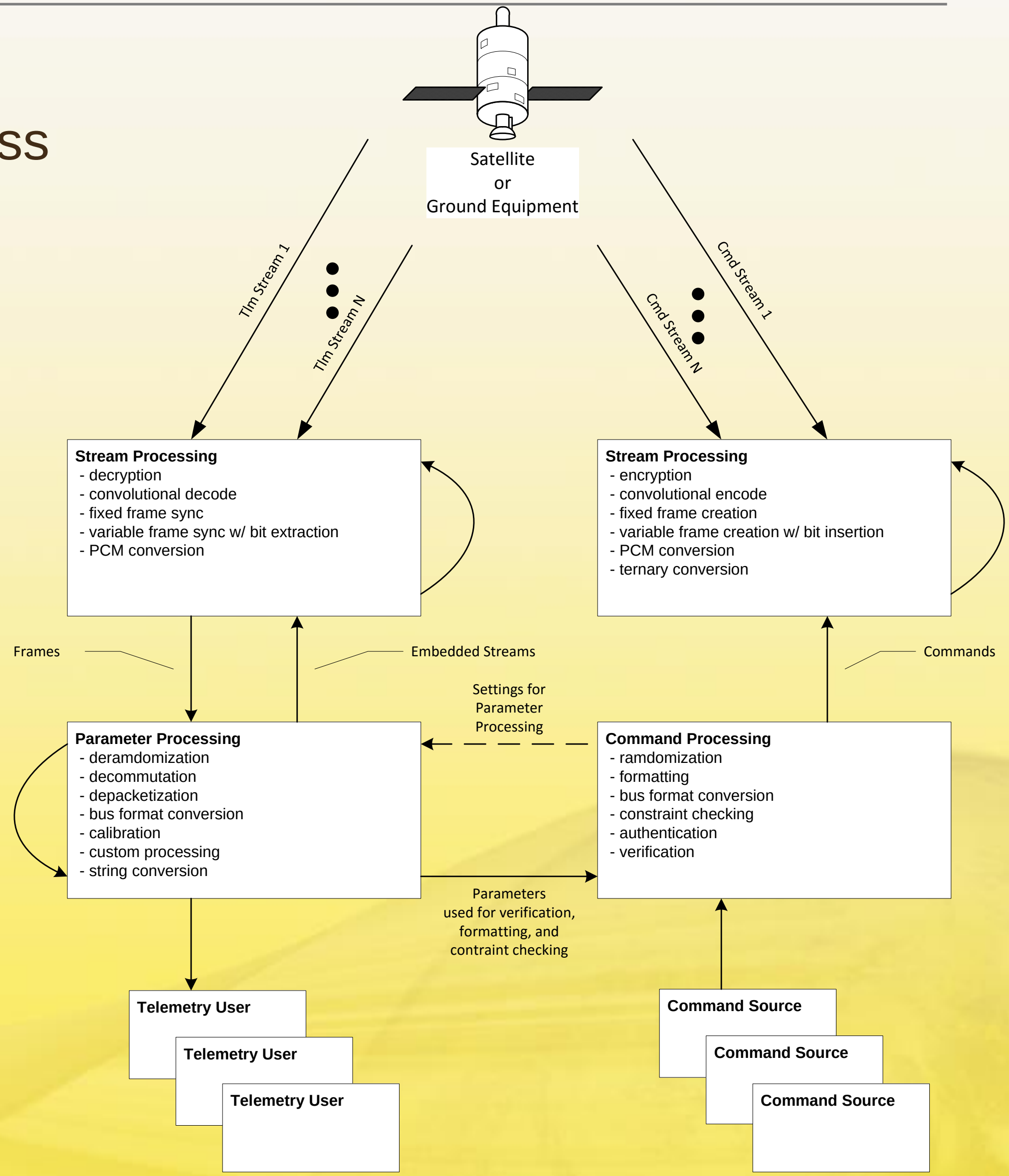
KRATOS®

Published by The Aerospace Corporation with permission.

OBJECT MANAGEMENT GROUP

XTCE in a Nutshell

- XTCE contains all the information required to process telemetry and command a spacecraft



Stream Processing
- decryption
- convolutional decode
- fixed frame sync
- variable frame sync w/ bit extraction
- PCM conversion

Stream Processing
- encryption
- convolutional encode
- fixed frame creation
- variable frame creation w/ bit insertion
- PCM conversion
- ternary conversion

Parameter Processing
- deramdomization
- decommutation
- depacketization
- bus format conversion
- calibration
- custom processing
- string conversion

Command Processing
- ramdomization
- formatting
- bus format conversion
- constraint checking
- authentication
- verification

Telemetry User

Telemetry User

Telemetry User

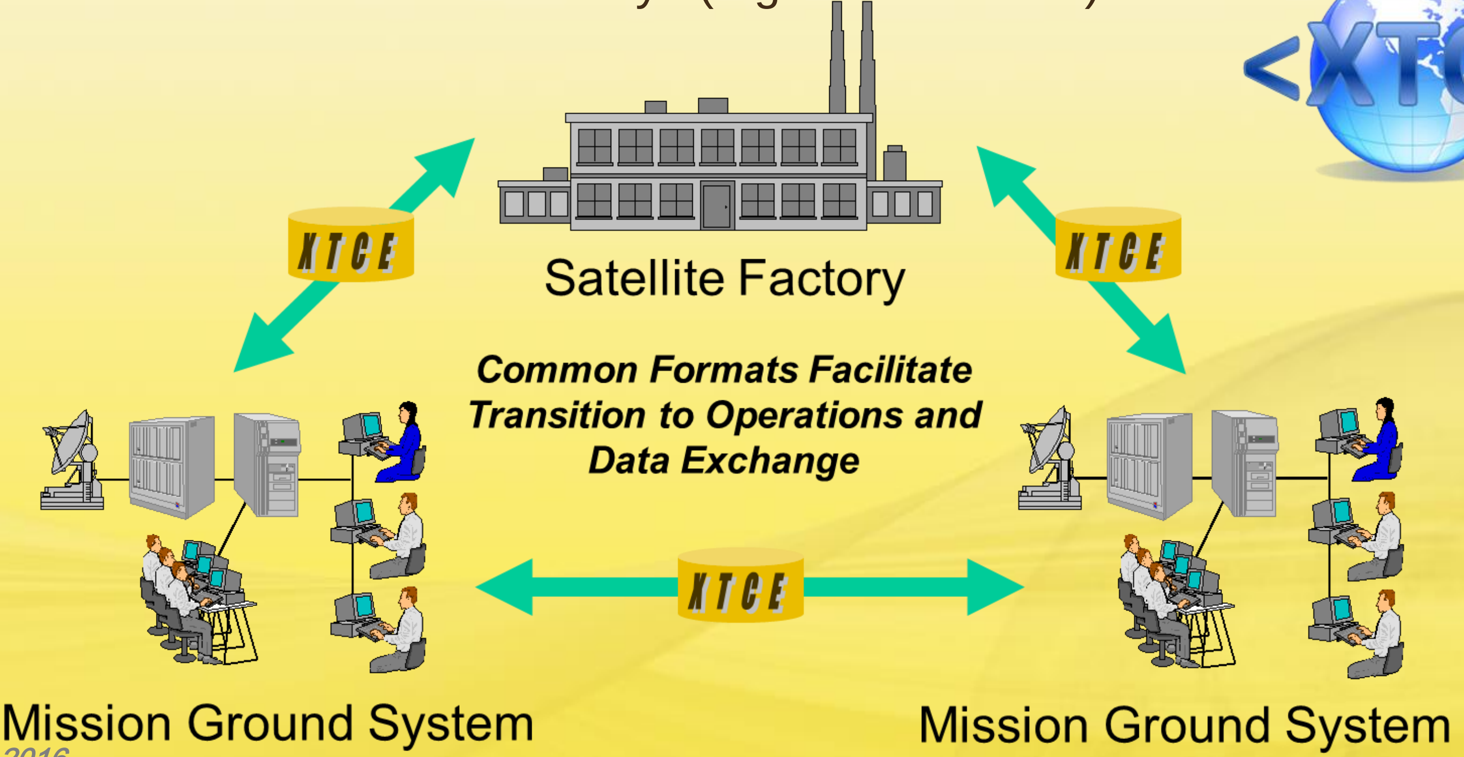
Command Source

Command Source

Command Source

What is XTCE?

- Standard for describing telemetry and commanding
- Lower cost and increase validation (XML) over traditional formats
- Conceived as universal exchange mechanism between ground segment apps (users) that need TLM/CMD descriptions
  - – But could be used in other ways (e.g. native format)



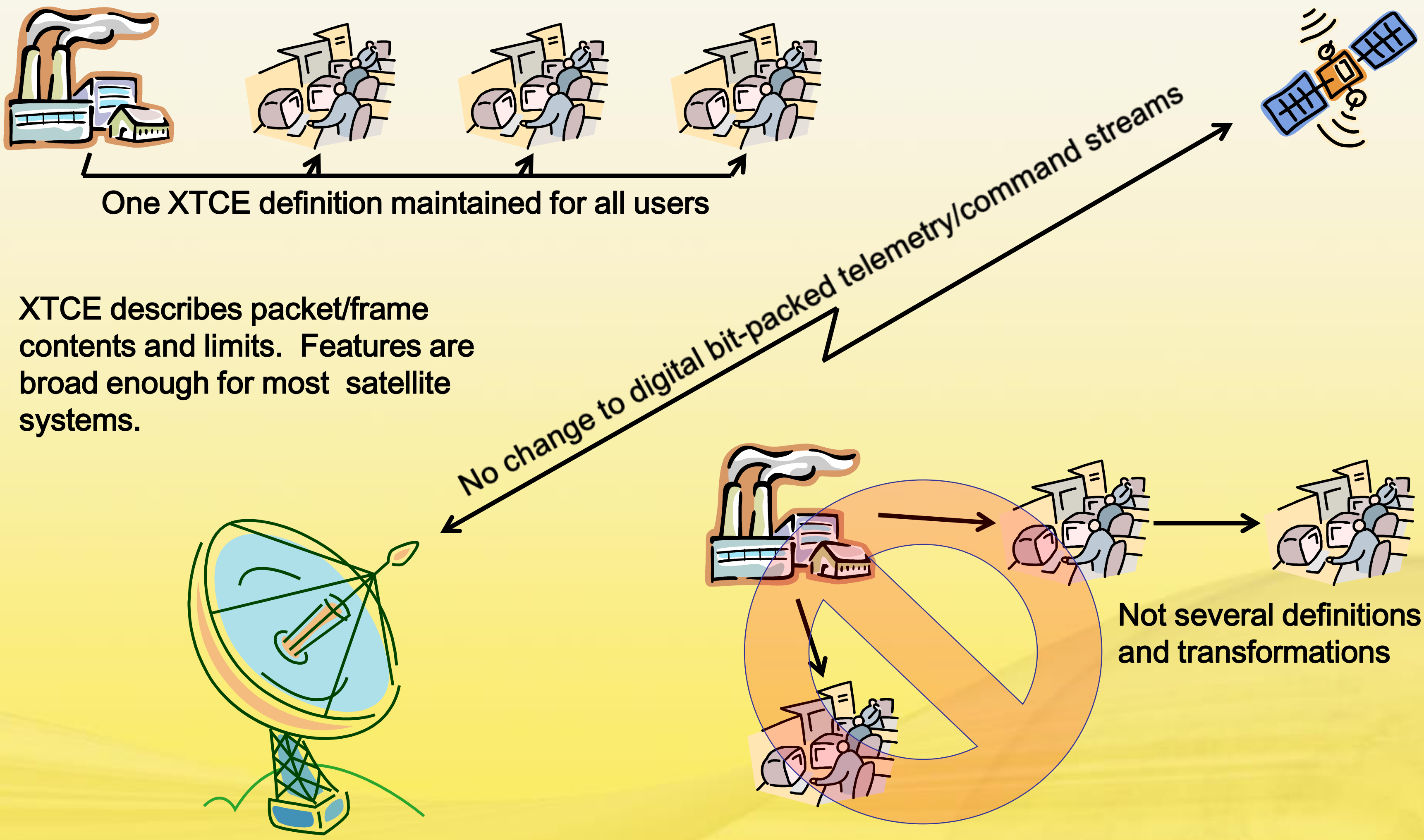
Satellite Factory

Common Formats Facilitate Transition to Operations and Data Exchange

Mission Ground System

Mission Ground System

Preface - Applicability



One XTCE definition maintained for all users

No change to digital bit-packed telemetry/command streams

XTCE describes packet/frame contents and limits. Features are broad enough for most satellite systems.

Not several definitions and transformations

Brief History and Background

- XTCE 1.0 published in 2005 (OMG only)
- XTCE 1.1 published in Oct 2007 (CCSDS and OMG)
- XTCE 1.1 Green Book – July 2006
- XTCE 1.2 Revision In-Progress at OMG.

Data → Meta Data → Meta Meta Data

Data	Description
W3C - Schema	Meta Meta Meta Data – An XML Documents (W3C Schema Language) that describes the format of itself
XTCE – Schema	Meta Meta Data – An XML Document (W3C Schema Language) that describes the XTCE format
XML in XTCE format	Meta Data – An XML Document (XTCE format) that describes the Data
Streams, Containers, Parameters & Commands	Data – Actual streams of bits containing packages of parameters or commands

Created by the W3C

Created by the SDTF

Created for a S/C program

Created by the spacecraft, Ground System or Simulator

Contents

- Preface
- The SpaceSystem
- XTCE Data Types
- Containers
- Describing Telemetry
- Alarms
- Commanding
- Trivial Sat
- Nuances

Basics

29 Feb 2016

Schema Nomenclature

- ‘Meta’ – means description. Example MetaCommand is a description of a Command.
- ‘Set’ – is an unordered collection
- ‘List’ – is an ordered collection
- ‘Ref’ – is a reference (by name) to an object defined elsewhere in the XML document

The SpaceSystem

The SpaceSystem – XTCE root

- The SpaceSystem is the root or the outermost element of any XTCE document

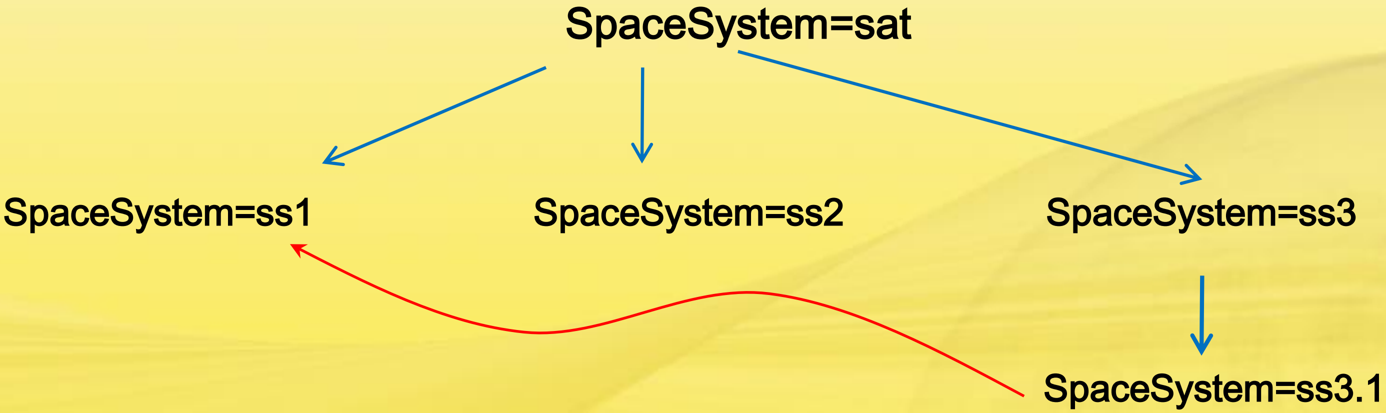
- Can be used to represent almost any aspect of your architecture
  - A single spacecraft
  - A spacecraft sub-system
  - An constellation of spacecraft

- The optional child SpaceSystem provides the ability to build a tree like structure of SpaceSystems

```
<xtce:SpaceSystem attributes>

    <xtce:TelemetryMetaData/>

    <xtce:CommandMetaData/>

    <xtce:SpaceSystem attributes/>

</xtce:SpaceSystem>
```

The SpaceSystem - Referencing

- Any <SpaceSystem> tree-structure is possible
  - Items defined in one <SpaceSystem> may refer to another using a named reference (“Ref”)
  - “Refs” come in various forms, similar syntax to directory paths
  - The simplest form is just the name of the item of interest, means local to current <SpaceSystem> or any above it
  - Absolute and relative path are also supported and go to the item the <SpaceSystem> specified
    - Absolute paths are **not** recommended (because they limit embedding one XTCE inside another)
- A single <SpaceSystem> with globally unique names of things is simplest



The SpaceSystem – Telemetry & Command Metadata

- Each SpaceSystem has a telemetry and command description area
  - Both are optional - a minimally valid XTCE file is a single SpaceSystem and name
  - TelemetryMetaData – describes telemetry items (mnemonics, packets, minor frames, etc…)
  - CommandMetaData – describes command items (arguments, cmds, cmd packets, etc…)
  - The name-space for TelemetryMetaData and CommandMetaData is shared
    - For example a Parameter defined in TelemetryMetaData is visible in CommandMetaData
      - It’s perfectly legal to refer to an item across Command & Telemetry Metadata

```
<xtce:SpaceSystem  name=“Empty Sat”>
    <xtce:TelemetryMetaData/>
    <xtce:CommandMetaData/>
</xtce:SpaceSystem>
```

The SpaceSystem – a NameDescriptionType

- A SpaceSystem is a collection of these major objects: ParameterTypes, Parameters, SequenceContainers, AurgumentTypes, Arguments, MetaCommands, CommandSequenceContainers and … more SpaceSystems

- Each of the above Object Types (which are NamedDescriptionTypes) carry additional descriptive data
  - shortDescription – this attribute is intended for short “tool tip” size descriptions
    - *Recommended* to be less than 80 characters (not enforced)
  - LongDescription – this sub-element is intended for “instructive” type descriptions. HTML markup is allowed in LongDescriptions and there is no length restriction.
  - AliasSet – Holds an optional set of alternative names for the objec, each associated with a NameSpace
    - e.g. AFSCN IRON, NORAD spacecraft number, or CCSDS SCID
  - AncillaryDataSet – Used to hold data that doesn’t otherwise fit in the schema
    - not exchange friendly

The SpaceSystem - Header

SpaceSystems may also contain a Header

This is strictly informational data

```
<Header version="1.8" date="Jul 20, 2006" classification="Unclassified"
validationStatus="Working">
    <AuthorSet>
        <Author> OMG Space DTF Team</Author>
        <Author> Gerry Simon</Author>
    </AuthorSet>
    <NoteSet>
        <Note> resemblance to an actual satellite is purely
coincidental</Note>
    </NoteSet>
    <HistorySet>
        <History>30 Nov, 2003 - Fully updated to release candidate
schema</History>
        <History>20 Jul, 2006 - Now updated for version 1.1</History>
    </HistorySet>
</Header>
```

XTCE Data Types

29 Feb 2016

Data Types – Base Types

- ParameterTypes and Command ArgumentsTypes have the same family of data types
  - **Integer** (32, 64 or 128 bit)
  - **Float** (32, 64 or 128 bit)
  - **Binary** (any size)
  - **String** (any size)
  - **Enumerated** – has a list of numerical values each with a matching string
  - **Boolean** – a two valued enumerated type
  - **RelativeTime** – an elapsed time value (precision is not specified)
  - **AbsoluteTime** – time referenced to a fixed epoch
  - **Array** – an array of some type
  - **Aggregate** – a grouping of members each with a Data Type
- Similar to those found in almost any programming language
- ParameterTypes are instantiated as Parameters. Parameters can have a value; a Parameter is not the value itself
- XTCE Types say nothing about how a Parameter or Command Argument is encoded for transmission
  - more on that next

Data Types – Instantiation

- Parameters are instantiations of ParameterTypes

```
<FloatParameterType name="BatteryVoltageType">
        <UnitSet/>
</FloatParameterType>
<Parameter name="BatteryVoltage1" parameterTypeRef="BatteryVoltageType"/>
<Parameter name="BatteryVoltage2" parameterTypeRef="BatteryVoltageType"/>
```

- Command Arguments are instantiations of ArgumentTypes

```
<BinaryArgumentType name="opCodeType">
        <UnitSet/>
</BinaryArgumentType>
<Argument name="opCode" argumentTypeRef="opCodeType"/>
```

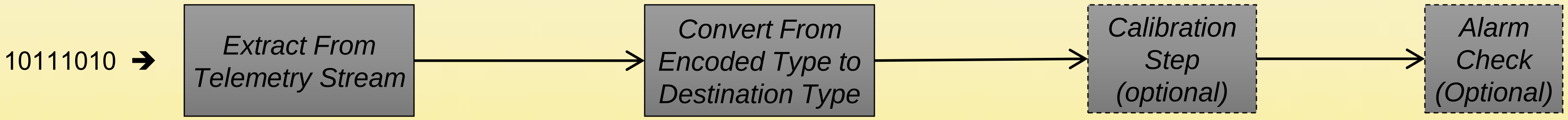
- Parameters and Command Arguments can be given an initial value

Data Types – Data Encoding

- Data Types (either Parameter or Argument) will usually specify how that data is to be encoded for transmission

- There are four encoding data types:
  - StringDataEncoding
    - UTF8 or UTF16 encoded Unicode (UTF-16 big endian)
    - Bit size, various ways
  - IntegerDataEncoding
    - Unsigned or TwosComplement (spelled “Compliment” in XTCE!)
    - Bit size
    - bit/byte order
    - Various calibrators such as Linear Calibrator or Polynomial Calibrator optional
  - FloatDataEncoding
    - IEEE-754, MIL-STD-1750A
    - Bit size
    - bit/byte order
    - Various calibrators such as Linear Calibrator or Polynomial Calibrator optional
  - BinaryDataEncoding
    - Bit size
    - bit/byte order
    - No Calibration

Data Types – Parameter Processing



10111010 ➔

Extract From Telemetry Stream

Convert From Encoded Type to Destination Type

Calibration Step (optional)

Alarm Check (Optional)

__DataEncoding__

-StringDataEncoding

-IntegerDataEncoding

-FloatDataEncoding

-BinaryDataEncoding

__ParameterType__

-String

-Float

-Integer

-Enumeration

-Binary

-Boolean

-Time

-Array, Aggregate

Data Types – Likely Data Type/DataEncoding Combinations

ParameterType	DataEncoding	Result
StringParameterType	StringDataEncoding	Unicode String encoded as either UTF-8 or UTF-16
EnumeratedParameterType	IntegerDataEncoding	LABEL, VALUE pairs
BinaryParameterType	BinaryParameterType	Blob data
IntegerParameterType	IntegerDataEncoding	• Integers of various well known formats (one/twos complement, etc…) • Calibrated/uncalibrated an option
FloatParameterType	FloatDataEncoding	•Floats of well known formats (IEEE/MIL1750A) • Calibrated/uncalibrated an optiona
FloatParameterType	IntegerDataEncoding	•Integer counts to units conversion • Calibrated
BooleanParameterType	IntegerDataEncoding	True/False
AbsoluteTimeParameterType	IntegerDataEncoding	•Time, same for RelativeTimeParameterType
Any of the above	BinaryDataEncoding	Format not describable with what is there
Aggregate or Array ParameterType	N/A	These NameReference other ParameterTypes

Data Types - Calibrators

- Calibrators are defined in the DataEncoding area
- A DefaultCalibrator
  - Just one
- Or ContextCalibrators
  - Varied
  - Conditional “Contexts”, user defined (e.g. Mission Phases)
- We’ll look at two common types
  - polynomial (PolynomialCalibrator)
  - line segment (SplineCalibrator)

Data Types - Linear Calibrator

```
<xtce:SplineCalibrator>
    <!--Forward (regular) calibration   -->
    <xtce:SplinePoint  raw="1" calibrated="10"/>
    <xtce:SplinePoint  raw="2" calibrated="100"/>
    <xtce:SplinePoint  raw="3" calibrated="500"/>
</xtce:SplineCalibrator>
```

There’s an optional @order attribute in SplinePoint – this is an error in the Schema, ignore

Data Types - Polynomial Calibrator

The equation for the calibration is: $y = -\,0.0048x^2 + 1.4091x - 48.886$

```
<xtce:PolynomialCalibrator>
    <!--Forward (regular) calibration   -->
    <xtce:Term exponent=“0" coefficient="-48.886"/>
    <xtce:Term exponent="1" coefficient="1.4091"/>
    <xtce:Term exponent=“2" coefficient="-0.0048"/>
</xtce:PolynomialCalibrator>
```

$-\,48.886$

$1.4091x$

$-\,0.0048x^2$

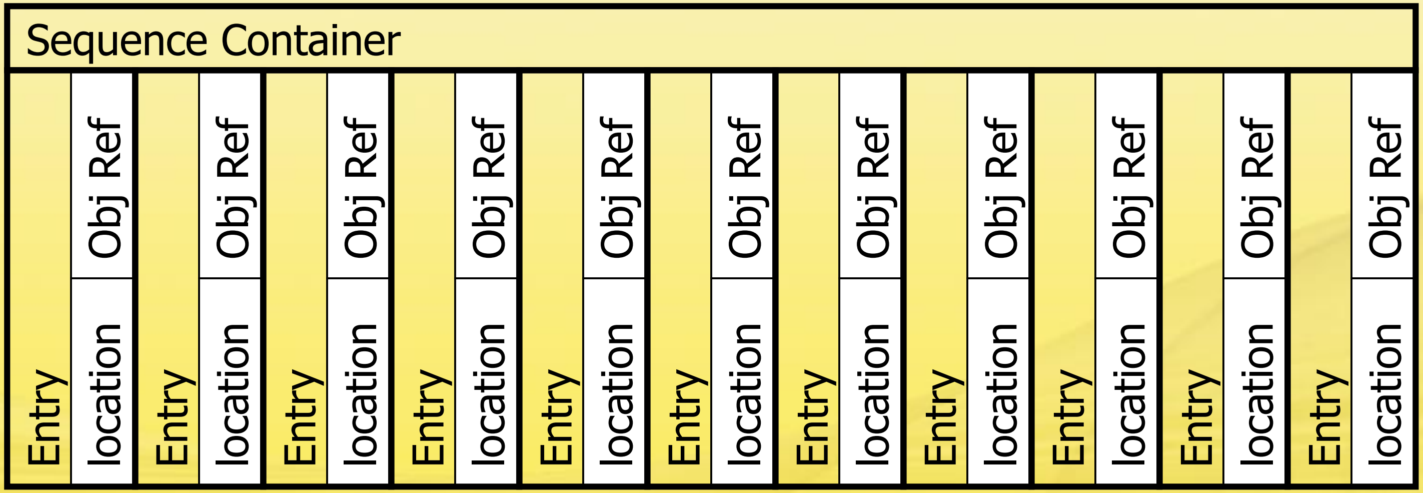
The number of terms is unlimited in the Schema, though most ground systems have a finite limit

Containers

29 Feb 2016

Containers – Sequence Container

- Has a list of Entry’s. Entry’s may be a reference to a Parameter (including arrays), a Parameter segment, a Stream segment, a Container, a Container segment, or an indirect reference to a Parameter.

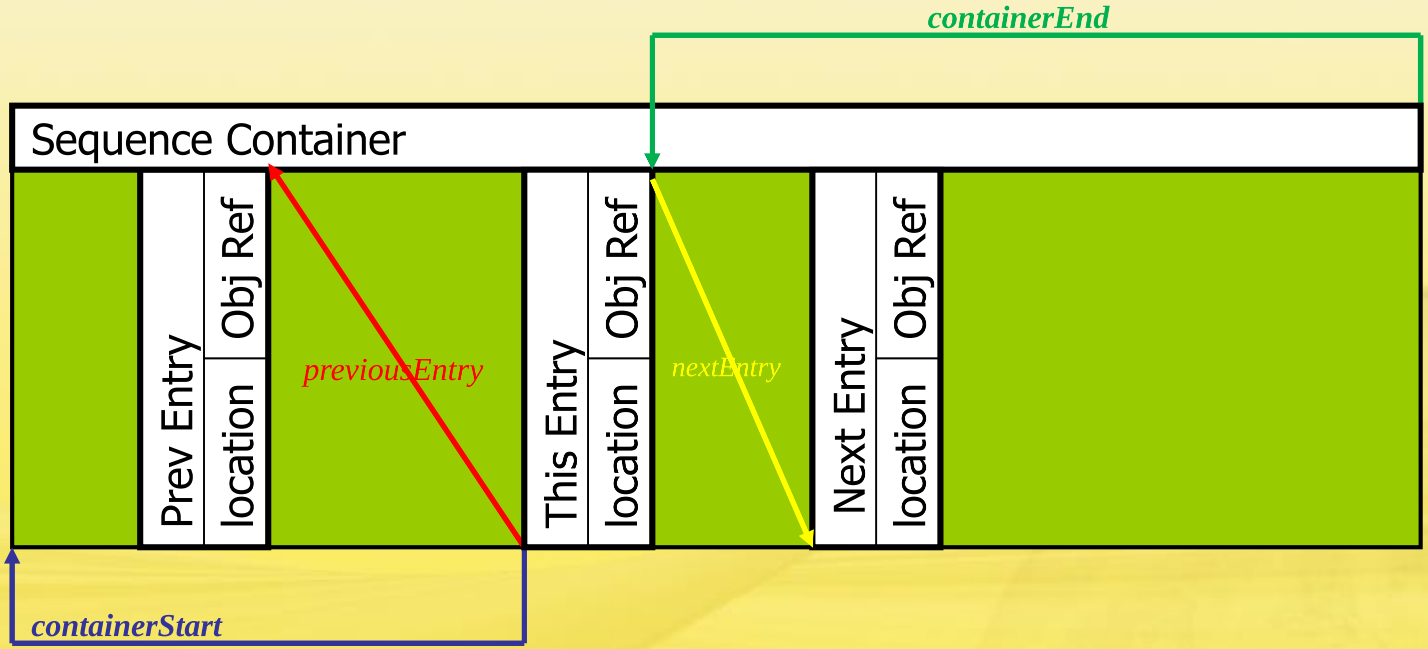


Sequence Container

Entry | location | Obj Ref
Entry | location | Obj Ref
Entry | location | Obj Ref
Entry | location | Obj Ref
Entry | location | Obj Ref
Entry | location | Obj Ref
Entry | location | Obj Ref
Entry | location | Obj Ref
Entry | location | Obj Ref
Entry | location | Obj Ref
Entry | location | Obj Ref

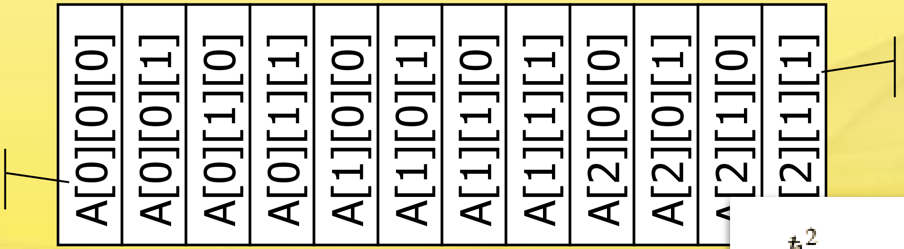
Containers – Sequence Container Location

- Location of the entry is an integer value referenced from:
  - The end of the previous entry to the start of This Entry (**default**)
  - The start of the container to the start of This Entry (containerStart)
  - The end of the container to the end of This Entry (containerEnd)
  - The start of the next entry to the end? of this entry (nextEntry)



Containers– Array Entries

- An Array entry may refer to the entire array, a part of the array or a single element of the array.
- Entire Array: must supply the size of every dimension (e.g. A[2][5][3]). Array order is assumed to have a sequence where the 0th elements in the first (most significant) dimension is supplied first.
  - Thus, a three dimensional array whose size is A[3][2][2] will have the sequence order as shown below



Most significant dimension

Least significant dimension

$$\frac{-\hbar^2}{2m}\nabla^2\Psi + U(x,y,z)\Psi(x,y,z) = E\Psi(x,y,z)$$

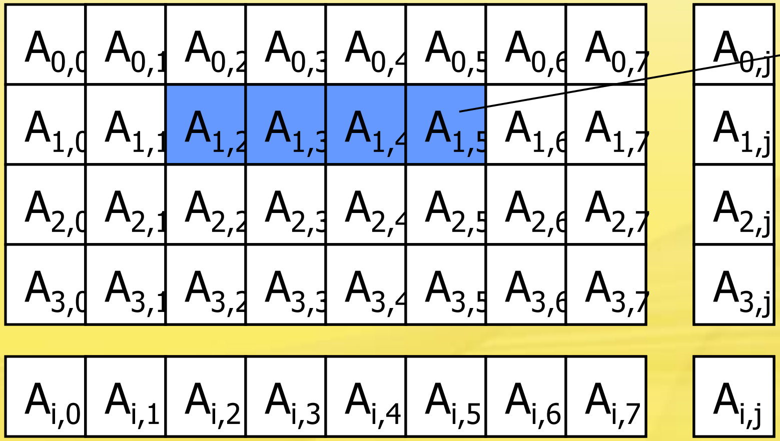
Containers – Array Entries

- Partial Array: Where the entry is only part of the array (e.g. a row) and may simply be a single element of the array.

- Must supply the starting index and ending index for each dimension to be included in the entry.

$$\frac{-\hbar^2}{2m}\nabla^2\Psi + U(x,y,z)\Psi(x,y,z) = E\Psi(x,y,z)$$

Array $A_{i,j}$



$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$	$A_{0,4}$	$A_{0,5}$	$A_{0,6}$	$A_{0,7}$		$A_{0,j}$
$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{1,4}$	$A_{1,5}$	$A_{1,6}$	$A_{1,7}$		$A_{1,j}$
$A_{2,0}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	$A_{2,4}$	$A_{2,5}$	$A_{2,6}$	$A_{2,7}$		$A_{2,j}$
$A_{3,0}$	$A_{3,1}$	$A_{3,2}$	$A_{3,3}$	$A_{3,4}$	$A_{3,5}$	$A_{3,6}$	$A_{3,7}$		$A_{3,j}$
$A_{i,0}$	$A_{i,1}$	$A_{i,2}$	$A_{i,3}$	$A_{i,4}$	$A_{i,5}$	$A_{i,6}$	$A_{i,7}$		$A_{i,j}$

Entry, size of least significant (last) dimension is 8 starting index for the last dimension is 2 and the ending index for the first dimension is 5. The starting index for the most significant (first) dimension is 1 and the ending index is 1.

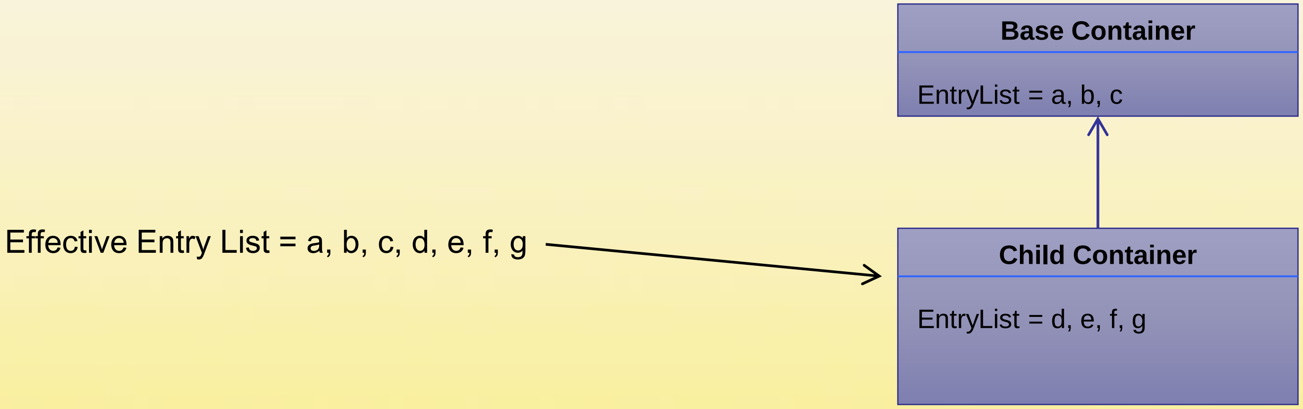
Containers - Entry Complexities

rarely used

- Location may be provided dynamically, from the value of a Parameter Instance (e.g., the start of a packet from in a CCSDS transfer frame).
- May have an “include” condition – A condition where the entry will not be included in the sequence.
- May include repeating contents where the count is given as a Parameter Instance value.

Containers - Inheritance

- SequenceContainers are built up as ‘type’ trees using inheritance (generalization/specialization) techniques borrowed from the Object Oriented regimen



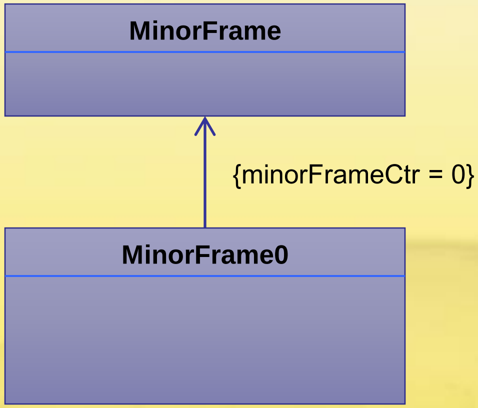
- The EntryList of the Base container is inherited by the child container and the Child’s EntryList is concatenated to that of the Base.
- Doubly defined space is OK, multiple overlapping parameters are assumed.

Containers – Inheritance, Setting the Base Container & Restrictions

- The element <BaseContainer> points to the parent container and the <RestrictionCriteria> element is usually a simple comparison that when it tests true the Parent Container is a type of Child

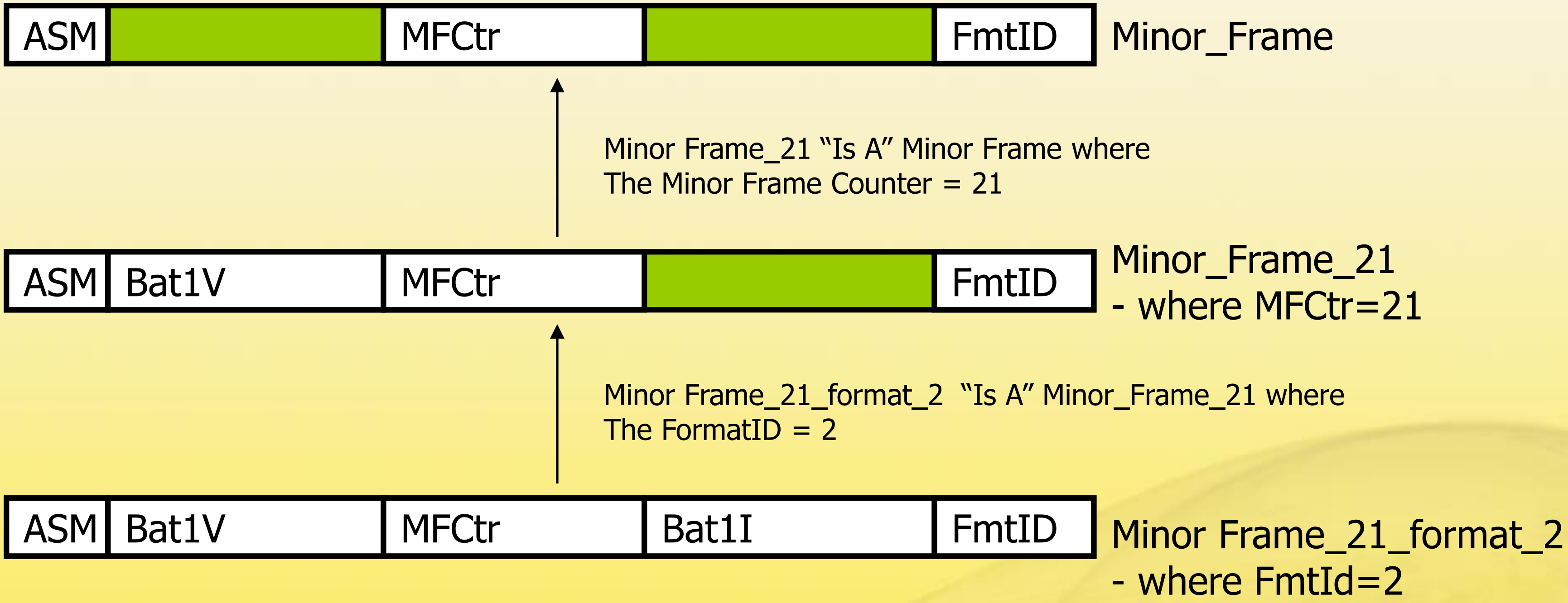
```
<SequenceContainer name="MinorFrame0">
      <EntryList> <!-- unimportant details --></EntryList>
      <BaseContainer containerRef="MinorFrame">
            <RestrictionCriteria>
                  <Comparison parameterRef="minorFrameCtr“
            </RestrictionCriteria>
      </BaseContainer>
</SequenceContainer>
```

Important



  - Should’ve been called ‘Constraints’

Containers – Sequence Container Inheritance Example



ASM 🟩 MFCtr 🟩 FmtID Minor_Frame

Minor Frame_21 “Is A” Minor Frame where
The Minor Frame Counter = 21

ASM Bat1V MFCtr 🟩 FmtID Minor_Frame_21
- where MFCtr=21

Minor Frame_21_format_2 “Is A” Minor_Frame_21 where
The FormatID = 2

ASM Bat1V MFCtr Bat1I FmtID Minor Frame_21_format_2
- where FmtId=2

Describing Telemetry

29 Feb 2016

Describing Telemetry – XML

```
<xtce:SpaceSystem>
    <xtce:TelemetryMetaData>
        <xtce:ParameterTypeSet/>
        <xtce:ParameterSet>
    </xtce:TelemetryMetaData>
    <xtce:CommandMetaData/>
    <xtce:SpaceSystem/>
</xtce:SpaceSystem>
```

Describing Telemetry - ParameterTypeSet

- The ParameterTypeSet holds the definition of all the parameters of interest for this SpaceSystem. ParameterTypes can be anything from header information, engineering values to sensor info, etc...
- Parameter types: Integer, Float, String, Enumeration, Binary, Absolute Time, Relative Time, Array, Aggregate and Boolean
- Simple Example

```
<IntegerParameterType name="grdCommandCtr">
   <UnitSet/>
</IntegerParameterType>
```

Describing Telemetry - ParameterType Examples

IntegerParameterType

```
<xtce:IntegerParameterType signed="false" name=“MySampleType">
  <xtce:UnitSet/>
  <xtce:IntegerDataEncoding sizeInBits="32"/>
  <xtce:DefaultAlarm>
     <xtce:StaticAlarmRanges>
       <xtce:WatchRange minInclusive="-2000.0" maxInclusive="5000.0"/>
       <xtce:WarningRange minInclusive="-5000.0" maxInclusive="6000.0"/>
     </xtce:StaticAlarmRanges>
   </xtce:DefaultAlarm>
</xtce:IntegerParameterType>
```

Describing Telemetry - ParameterType Examples

FloatParameterType

```
<xtce:FloatParameterType sizeInBits="64" name=“DecaySensorType">
  <xtce:UnitSet>
     <xtce:Unit description="Bq">Becquerel</xtce:Unit>
  </xtce:UnitSet>
  <xtce:IntegerDataEncoding sizeInBits="16" encoding="twosCompliment">
    <xtce:DefaultCalibrator>
       <xtce:SplineCalibrator>
           <xtce:SplinePoint raw="-32768.0" calibrated="0.0"/>
           <xtce:SplinePoint raw="0.0" calibrated="5.0"/>
          <xtce:SplinePoint raw="32767.0" calibrated="20.0"/>
       </xtce:SplineCalibrator>
    </xtce:DefaultCalibrator>
  </xtce:IntegerDataEncoding>
</xtce:FloatParameterType>
```

Describing Telemetry - Parameter Set

- Hold all the Parameters in the SpaceSystem
  - Parameters instantiate Parameter Types
- Each Parameter may have an initial value & Parameter Properties
- The ParameterSet may also have a ParameterReference
  - Links to a Parameter defined somewhere else in the SpaceSystem tree

```
<ParameterSet>
    <Parameter name="frameSync" parameterTypeRef="frameSyncType">
        <ParameterProperties dataSource="telemetered" readOnly="true"/>
    </Parameter>
    <Parameter name="minorFrameCtr" parameterTypeRef="minorFrameCtrType"/>
    <Parameter name="grdCommandCtr" parameterTypeRef="grdCommandCtrType"/>
    <Parameter name="formatID" parameterTypeRef="formatIDType"/>
    <Parameter name="spacecraftID" parameterTypeRef="spacecraftIDType“ initialValue=“77”/>
    <Parameter parameterTypeRef="spacecraftTimeType" name="spacecraftTime"/>
    <Parameter parameterTypeRef="durationType" name="timeSinceLastLaunch"/>
</ParameterSet>
```

Describing Telemetry – Example XML

```
<BinaryParameterType name="frameSyncType" shortDescription="Frame Synchronization Field Type"
initialValue="0934">
    <LongDescription><![CDATA[This long description for frame sync contains an <b>HTML</b>markup
description]]></LongDescription>
    <UnitSet/>
    <BinaryDataEncoding bitOrder="mostSignificantBitFirst">
        <SizeInBits>
            <FixedDecimalValue>13</FixedDecimalValue>
        </SizeInBits>
    </BinaryDataEncoding>
</BinaryParameterType>


<Parameter name="frameSyncType" parameterTypeRef="frameSyncType" shortDescription="Frame Synchronization
        Field">
    <ParameterProperties dataSource="telemetered" readOnly="true"/>

</Parameter>
```

Note the embedded HTML in the long description

Describing Telemetry - Parameter Properties

- dataSource
  - Optional, one of: {telemetered, derived, constant, local}
- readOnly
  - Normally used for system variables that do not update
- SystemName
  - Optional. Normally used when the database is built in a flat, non-hierarchical format and the system name cannot be derived from the hierachy.
- ValidityCondition
  - Condition that must be true for this Parameter to be valid
- PhysicalAddressSet
  - Contains the address (e.g., channel information) required to process the spacecraft telemetry streams. May be an onboard id, a mux address, or a physical location.
- TimeAssociation
  - Helps to time tag telemetry - see discussion in the Nuances section

Describing Telemetry – Session or System Variables

- To define a Parameter used by the ground system
  - Simply leave off the DataEncoding in ParameterType
    - System variables are local
  - And then make a Parameter that refers to it
  - Session variables often seem to be useful in a Comparison, where the information is held outside the all the descriptions in the file…

```
<xtce:IntegerParameterType name="GroundCommand CtrType">
```

Describing Telemetry Formats – SequenceContainers

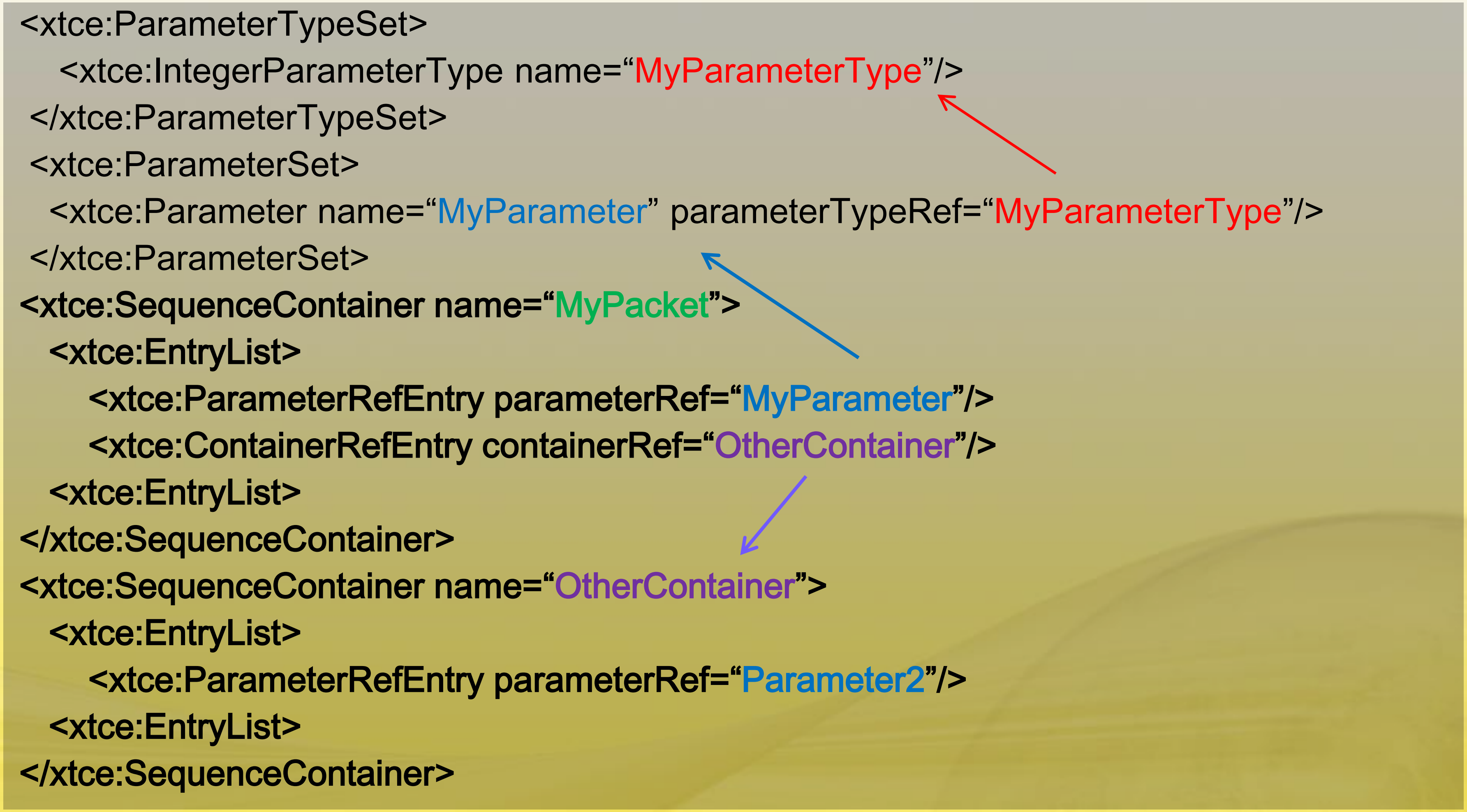
```
<xtce:SpaceSystem>
    <xtce:TelemetryMetaData>
        <xtce:ParameterTypeSet/>
        <xtce:ParameterSet/>
        <xtce:ContainerSet/>
    </xtce:TelemetryMetaData>
    <xtce:CommandMetaData/>
    <xtce:SpaceSystem/>
</xtce:SpaceSystem>
```

Describing Telemetry – SequenceContainers

- Use <SequenceContainer> to describe telemetry formats (CCSDS packet, etc…)
  - Specify Sequence of parameters
    - various options to manipulate sequence
  - Optionally EXTEND another container in an inheritance like fashion
    - Use element <BaseContainer> to name the parent container
    - Specify <RestrictionCriteria> to define identifying "keys" and expected values (such as APID=10)
    - Principally a construction mechanism, child inherits certain properties from the parent, mainly its entries
    - And also used to provide the identifying information
- (CommandContainers and MetaCommand/CommandContainers are similar)

```
<xtce:TelemetryMetaData>
    <xtce:ParameterSet/>
    <xtce:ParameterTypeSet/>
    <xtce:ContainerSet>
      <xtce:SequenceContainer/>
    </xtce:ContainerSet>
</xtce:TelemetryMetaData
```

Describing Telemetry Formats -Simple Example

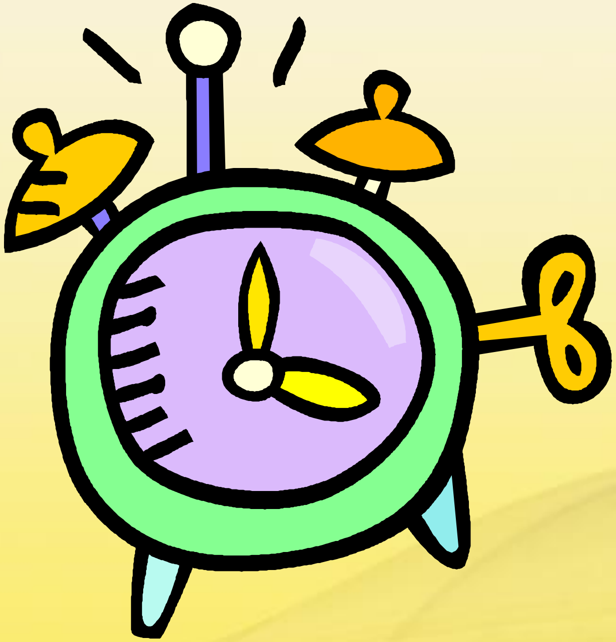


```
<xtce:ParameterTypeSet>
    <xtce:IntegerParameterType name="MyParameterType"/>
 </xtce:ParameterTypeSet>
 <xtce:ParameterSet>
   <xtce:Parameter name="MyParameter" parameterTypeRef="MyParameterType"/>
 </xtce:ParameterSet>
<xtce:SequenceContainer name="MyPacket">
   <xtce:EntryList>
       <xtce:ParameterRefEntry parameterRef="MyParameter"/>
       <xtce:ContainerRefEntry containerRef="OtherContainer"/>
   <xtce:EntryList>
</xtce:SequenceContainer>
<xtce:SequenceContainer name="OtherContainer">
   <xtce:EntryList>
       <xtce:ParameterRefEntry parameterRef="Parameter2"/>
   <xtce:EntryList>
</xtce:SequenceContainer>
```

Describing Telemetry Formats – Container Inheritance Recap

- Use Container Inheritance to add identifying “keys” to a full description
  - Keys are identifying locations in the bit-stream, like CCSDS application ID or minor frame number
    - You may have others
  - In Class diagrams they would be shown as “constraints
  - BaseContainer child element
    - Supply the Parent or “Super” Container Ref
    - Supply keys – a set of conditions that must be true for this description
      - Keys are usually Parameters in the Parent Container
      - And are probably IDENTIFYING areas of a packet/minor frame:
        - » CCSDS Application Identifier field in header
        - » TDM: Minor frame number

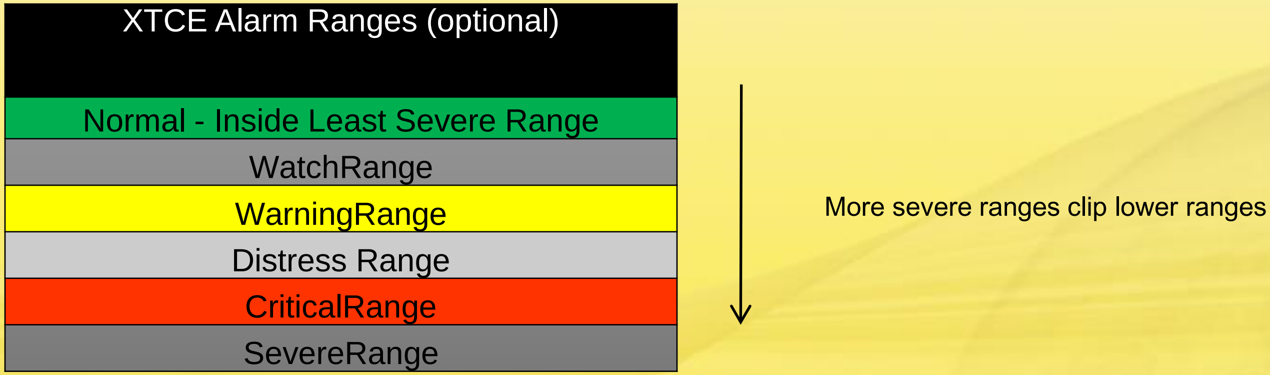
- Like class inheritance but more of a construction technique
  - You do get to say this container IS A that container
  - But instead of method overloading the EntryList of the parent is added to the child’s
  - And constraints (RestrictionCriteria) are usually interpreted as identifying keys in the bit-stream



Alarms

Alarms - Default and Context

- DefaultAlarm and ContextAlarms
  - Default is … well the default
  - Contexts – user defined such as mission phase
- A variety of limit checks are available in XTCE and tuned to their ParameterType somewhat:
  - AlarmConditions: check a condition
  - StaticAlarmRanges: compare values against set of ranges
  - ChangeAlarms: Rate or Delta
  - Slight variations for String & Enum alarms



XTCE Alarm Ranges (optional)

Normal - Inside Least Severe Range

WatchRange

WarningRange

Distress Range

CriticalRange

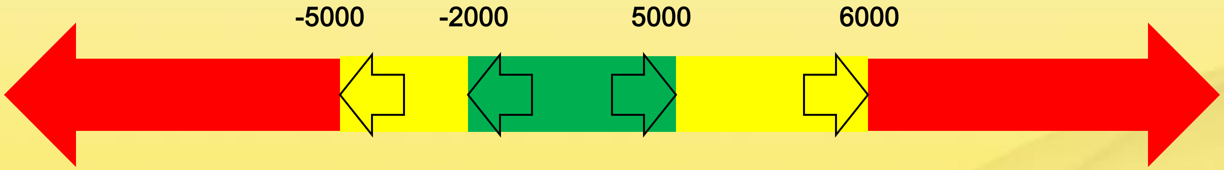
SevereRange

More severe ranges clip lower ranges

Alarms – StaticAlarmRange Example

```
<xtce:StaticAlarmRanges>
            <xtce:WatchRange minInclusive="-2000.0" maxInclusive="5000.0"/>
           <xtce:SevereRange minInclusive="-5000.0" maxInclusive="6000.0"/>
</xtce:StaticAlarmRanges>
```

Real Ranges:
-Green: -2000 .0 > x < 5000.0 – implied
-Watch: x <= -2000.0 or x >= 5000.0
-Severe: x <= -5000.0 or x >= 6000.0

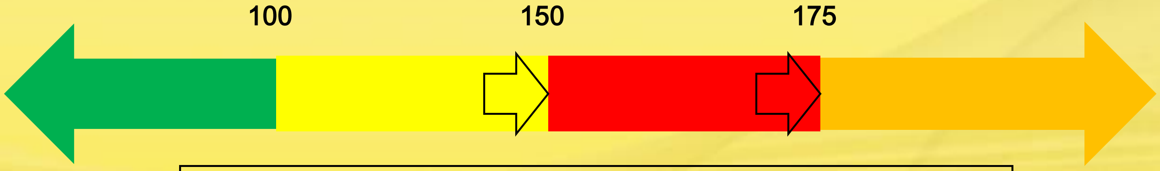


Note: Color designation is not part of XTCE, user dependent

Alarms - AlarmConditions

- Set up conditions for up to five levels increasing in severity:
  - normal -> watch -> warning -> distress -> critical -> severe
  - All of the ranges are optional
  - The normal range is implied to be inside the least severest range
  - The alarm “returns” the highest level triggered
  - Simple single conditions, multiple conditions, boolean expressions, custom algorithms

```
<xtce:AlarmConditions>
      <xtce:WarningAlarm>
            <xtce:Comparison parameterRef="pressureValue" value="100" comparisonOperator="&gt;"/>
      </xtce:WarningAlarm>
      <xtce:CriticalAlarm>
           <xtce:Comparison parameterRef="pressureValve" value="150" comparisonOperator="&gt;"/>
      </xtce:CriticalAlarm>
      <xtce:SevereAlarm>
         <xtce:Comparison parameterRef="pressureValve" value="175" comparisonOperator="&gt;"/>
      </xtce:SevereAlarm>
</xtce:AlarmConditions>
```



Note 1: Color designation is not part of XTCE, user dependent

Note 2: The meaning of the alarm ranges is not specified by XTCE

Alarms – ChangeAlarms

- Used to trigger alarms when the rate-of-change of a parameter’s value is either too fast or too slow.

- The change may be either:
  - with respect to time (rate alarms)
    - changeType, set to changePerSecond
  - or with respect to samples (delta alarms)
    - changeType, set to changePerSample

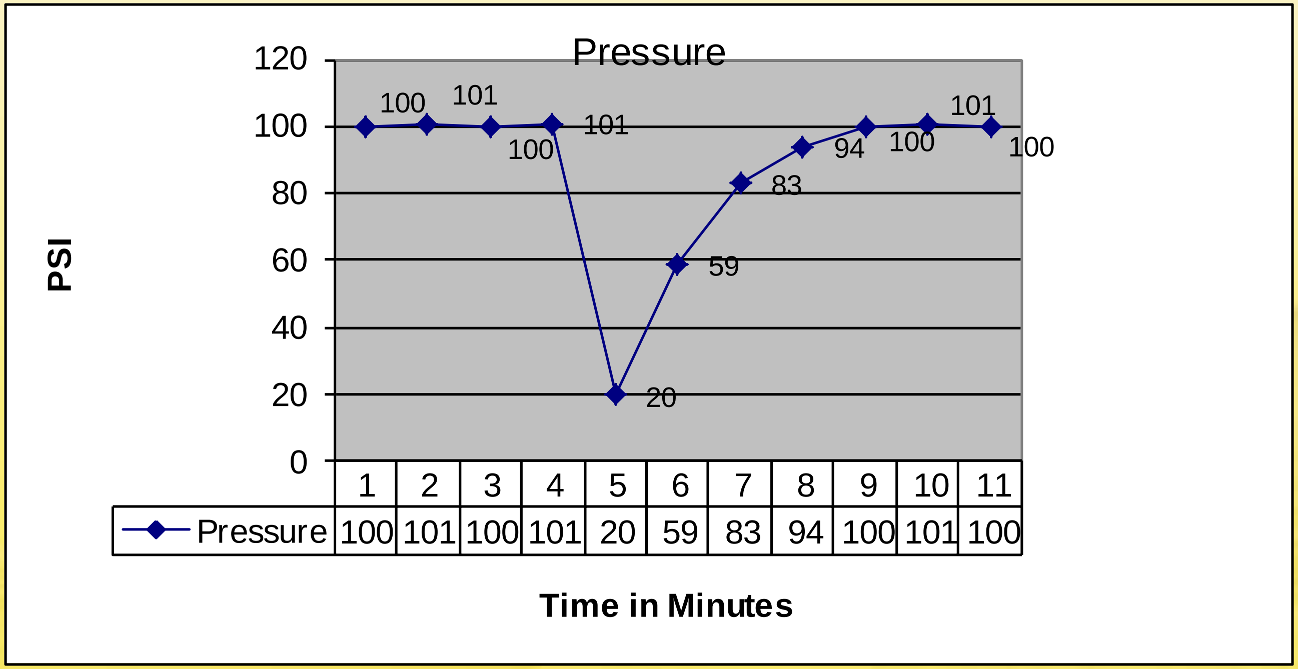
- The change may also be either:
  - relative
    - changeBasis, set to percentageChange
  - or absolute
    - changeBasis, set to absoluteChange

- To set period that must pass before a trigger
  - use spanOfInterestInSeconds – time (default is 0 i.e., check on every change)
  - use spanOfInterestInSamples – samples (default is 1 i.e., every sample)

Rate of Change		Delta Change	
Attribute	Value	Attribute	Value
@changeType	changePerSecond	@changeType	changePerSample
@changeBasis	absoluteChange | percentageChange	@changeBasis	absoluteChange | percentageChange
@spanOfInterestInSamples	ignore	@spanOfInterestInSamples	1 or more
@spanOfInterestInSeconds	1 or more	@spaceOfInterestInSeconds	ignore

Alarms – ChangeAlarms Tank Example

- Tank Pressure is sampled every minute
- Normal Pressure is 100 PSI, within a few percent
- ChangeAlarm used to determine if PSI change is outside of prescribed ranges
- This example shows a pressure release event followed by re-pressurization back to normal levels



	1	2	3	4	5	6	7	8	9	10	11
Pressure	100	101	100	101	20	59	83	94	100	101	100

Alarms - ContextAlarms

- Oftentimes Alarm ranges need to change as the SV enters different phases (factory test, on-orbit) or enters different operating modes (eclipse, LEO, safe-mode).
- These are all collectively called ‘Contexts’
- Parameters have an optional ContextAlarmList
- Alarms from the first Context in the ContextAlarmList to test true are applied.

```
<ContextAlarmList>
    <ContextAlarm>
        <StaticAlarmRanges>
            <WarningRange minInclusive="-22.2" maxExclusive="12.2"/>
            <CriticalRange minExclusive="-44" maxInclusive="14"/>
        </StaticAlarmRanges>
        <ContextMatch>
            <ComparisonList>
                <Comparison parameterRef="/sc-operating-environment" value="LEO"/>
            </ComparisonList>
        </ContextMatch>
    </ContextAlarm>
</ContextAlarmList>
```

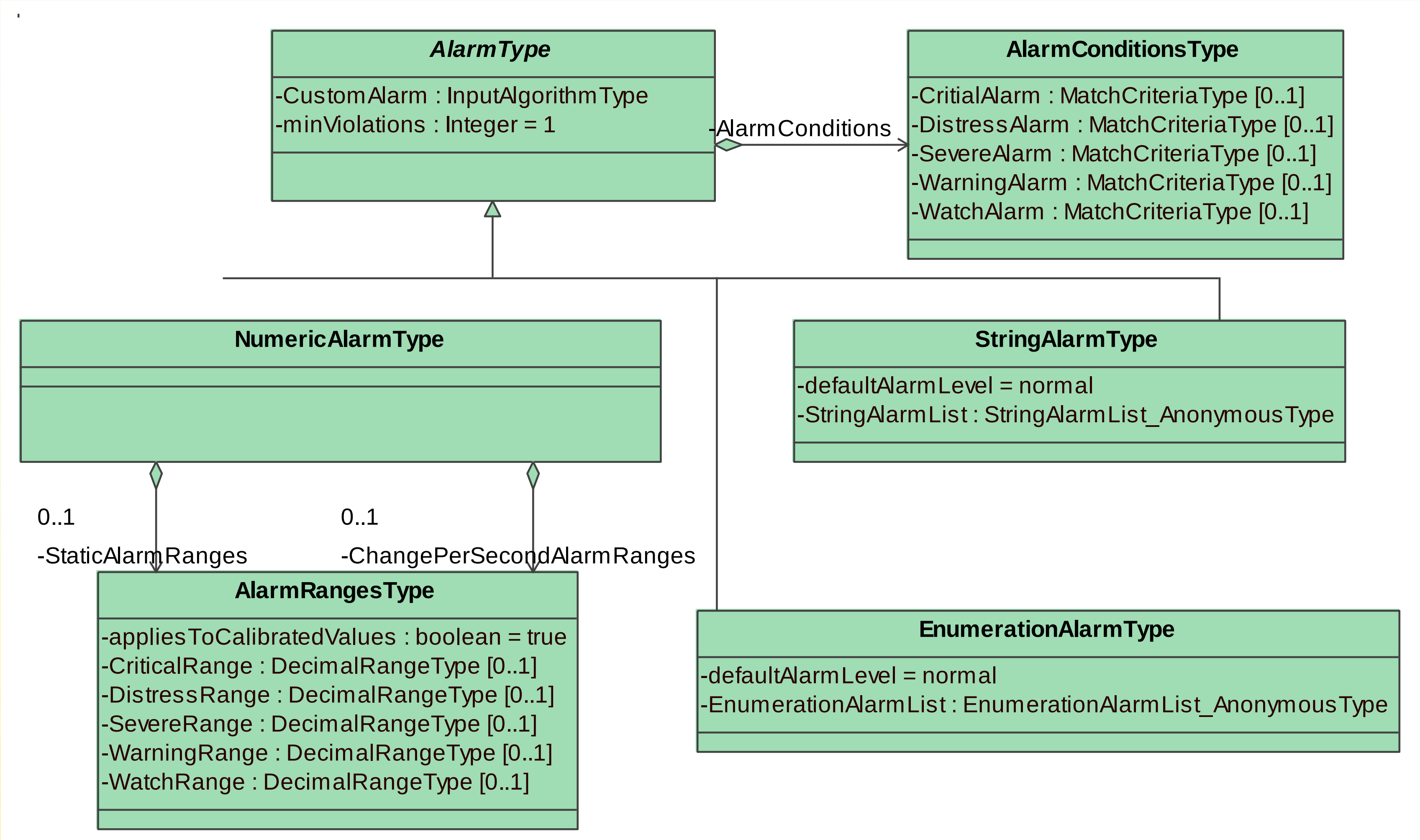
Alarms – Non Numeric (Less Common)

- Alarm Conditions
  - Used when the Alarm is more than a simple range
  - Alarm is set when a Condition is true
  - There is a Condition (MatchCriteria) for each Alarm level
  - Highest Level to test true will be the Alarm Condition
- Enumeration Alarms
  - Alarm levels associated with enumerated values
- String Alarms
  - Alarm levels associated with regular expressions

Alarms – User Algorithm

- Used when the conditions for the Alarm are too complex to otherwise describe in XTCE
- A reference to an external algorithm that will set the Alarm state

Alarms – Alarm simplified UML



AlarmType

-CustomAlarm : InputAlgorithmType
-minViolations : Integer = 1

-AlarmConditions

AlarmConditionsType

-CritialAlarm : MatchCriteriaType [0..1]
-DistressAlarm : MatchCriteriaType [0..1]
-SevereAlarm : MatchCriteriaType [0..1]
-WarningAlarm : MatchCriteriaType [0..1]
-WatchAlarm : MatchCriteriaType [0..1]

NumericAlarmType

StringAlarmType

-defaultAlarmLevel = normal
-StringAlarmList : StringAlarmList_AnonymousType

0..1
-StaticAlarmRanges

0..1
-ChangePerSecondAlarmRanges

AlarmRangesType

-appliesToCalibratedValues : boolean = true
-CriticalRange : DecimalRangeType [0..1]
-DistressRange : DecimalRangeType [0..1]
-SevereRange : DecimalRangeType [0..1]
-WarningRange : DecimalRangeType [0..1]
-WatchRange : DecimalRangeType [0..1]

EnumerationAlarmType

-defaultAlarmLevel = normal
-EnumerationAlarmList : EnumerationAlarmList_AnonymousType

Commanding

Commanding - XML

```xml
<xtce:SpaceSystem>
    <xtce:TelemetryMetaData/>
    <xtce:CommandMetaData>
       <xtce:ParameterSet/>
       <xtce:ParameterTypeSet/>
       <xtce:ArgumentTypeSet/>
       <xtce:MetaCommandSet/>
       <xtce:CommandContainerSet/>
    </xtce:CommandMetaData>
    <xtce:SpaceSystem/>
</xtce:SpaceSystem>
```

Commanding - Command ParameterTypes and ArgumentTypes

- ParameterTypes & Parameters defined in CommandMetaData are exactly the same and share the same name space as those defined in TelemetryMetaData
  - Can be defined here only as a convenience to the database author
- ArgumentTypes are similar in pattern to ParmeterTypes
  - But they lack alarms
  - They contain additional data for input verification
- MetaCommands contain descriptions of commands
  - The instantiation of Arguments from ArgumentType is local to MetaCommands
    - There is no ‘ArgumentSet’

Commanding – CommandMetaData Describing Commands

```
<xtce:SpaceSystem>
  <xtce:TelemetryMetaData>
      <xtce:ParameterTypeSet/>
      <xtce:ParameterSet/>
      <xtce:ContainerSet/>
  </xtce:TelemetryMetaData>
  <xtce:CommandMetaData>
      <xtce:ParameterTypeSet/>
      <xtce:ParameterSet/>
      <xtce:ArgumentTypeSet/>
      <xtce:MetaCommandSet>
            <xtce:MetaCommand>
                 <xtce:ArgumentList/>
                 <xtce:CommandContainer>
            </xtce:MetaCommand>
      </xtce:MetaCommandSet>
      <xtce:CommandContainerSet/>
    </xtce:CommandMetaData>
</xtce:SpaceSystem>
```

MetaCommand/CommandContainer has a LOCAL CommandContainer for Commands

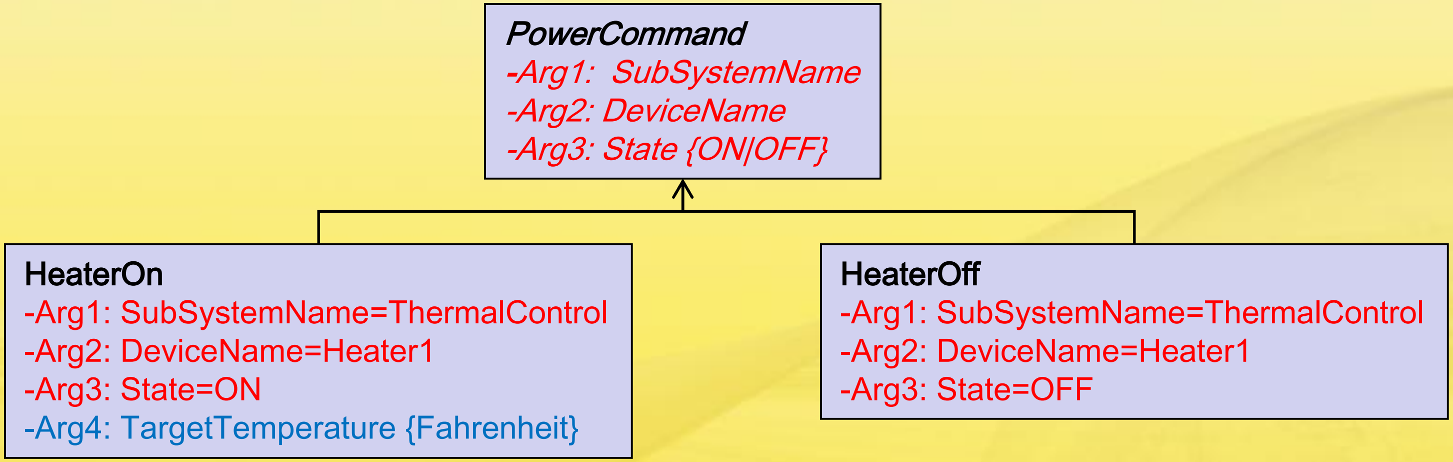
- Build Command Packets here!

Commanding – The MetaCommand Element

- ‘Data about Commands’
- MetaCommands may have Arguments for data that is supplied at run time (usually by an operator)
  - Verifiers, Interlock, various constraints, side effects
- MetaCommands may have a CommandContianer
  - CommandContainer is a SequenceContainer that may also have argument entries
  - The CommandContainer defines the packaging (order of arguments, fixed vales, parameters, etc.) for a MetaCommand
    - Used to build Command Packets or Command Blocks
  - Like SequenceContainers, CommandContainers may be derived from other CommandContainers via inheritance
- MetaCommands may be derived from other MetaCommands (via inheritance)

Commanding – MetaCommand Inheritance

- A MetaCommand may EXTEND another
  - Give the Ref of the MetaCommand being extended
- The extending MetaCommand can add arguments
  – It gets any Parent arguments and adds its own
- Or it can set arguments in the Parent’s Command
  – MetaCommand/BaseMetaCommand/ArgumentAssigmentList



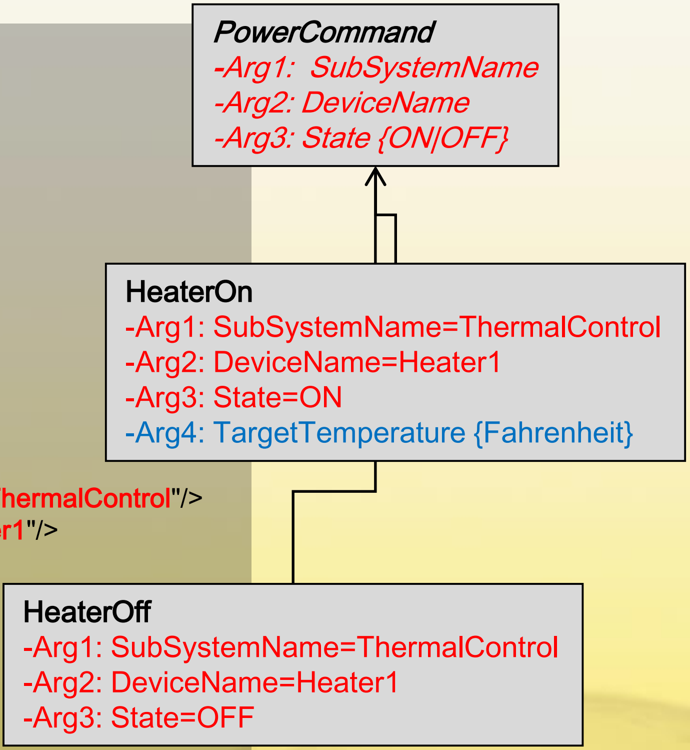
PowerCommand
-Arg1: SubSystemName
-Arg2: DeviceName
-Arg3: State {ON|OFF}

HeaterOn
-Arg1: SubSystemName=ThermalControl
-Arg2: DeviceName=Heater1
-Arg3: State=ON
-Arg4: TargetTemperature {Fahrenheit}

HeaterOff
-Arg1: SubSystemName=ThermalControl
-Arg2: DeviceName=Heater1
-Arg3: State=OFF

Commanding – MetaCommand Inheritance in XTCE

```
<xtce:MetaCommand name=“PowerCommand" abstract="true">
    <xtce:ArgumentList>
        <xtce:Argument name=“SubSystemName" argumentTypeRef=“SubSysEnumType"/>
        <xtce:Argument name=“DeviceName" argumentTypeRef=“DeviceEnumType"/>
        <xtce:Argument name=“State" argumentTypeRef=“StateEnumType"/>
   </xtce:ArgumentList>
   <xtce:CommandContainer/>  <!– details left out -->
</xtce:MetaCommand>
<xtce:MetaCommand name=“HeaterOn">
    <xtce:ArgumentList>
        <xtce:Argument name=“TargetTemperature argumentTypeRef=“FloatType"/>
          </xtce:ArgumentList>
          <xtce:BaseMetaCommand metaCommandRef=“PowerCommand”>
                <xtce:ArgumentAssignmentList>
                     <xtce:ArgumentAssignment argumentName="SubSystemName" argumentValue=“ThermalControl"/>
                     <xtce:ArgumentAssignment argumentName="DeviceName" argumentValue=“Heater1"/>
                     <xtce:ArgumentAssignment argumentName="State" argumentValue=“ON"/>
                </xtce:ArgumentAssignmentList>
         </xtce:BaseMetaCommand>
        <xtce:CommandContainer/>
</xtce:MetaCommand>
<xtce:MetaCommand name=“HeaterOff">
          <xtce:BaseMetaCommand metaCommandRef=“PowerCommand”/>
                <xtce:ArgumentAssignmentList>
                    <xtce:ArgumentAssignment argumentName="SubSystemName" argumentValue=“ThermalControl"/>
                    <xtce:ArgumentAssignment argumentName="DeviceName" argumentValue=“Heater1"/>
                    <xtce:ArgumentAssignment argumentName="State" argumentValue=“OFF"/>
                </xtce:ArgumentAssignmentList>
           </xtce:BaseMetaCommand>
           <xtce:CommandContainer/>
</xtce:MetaCommand>
```



PowerCommand
-Arg1: SubSystemName
-Arg2: DeviceName
-Arg3: State {ON|OFF}

HeaterOn
-Arg1: SubSystemName=ThermalControl
-Arg2: DeviceName=Heater1
-Arg3: State=ON
-Arg4: TargetTemperature {Fahrenheit}

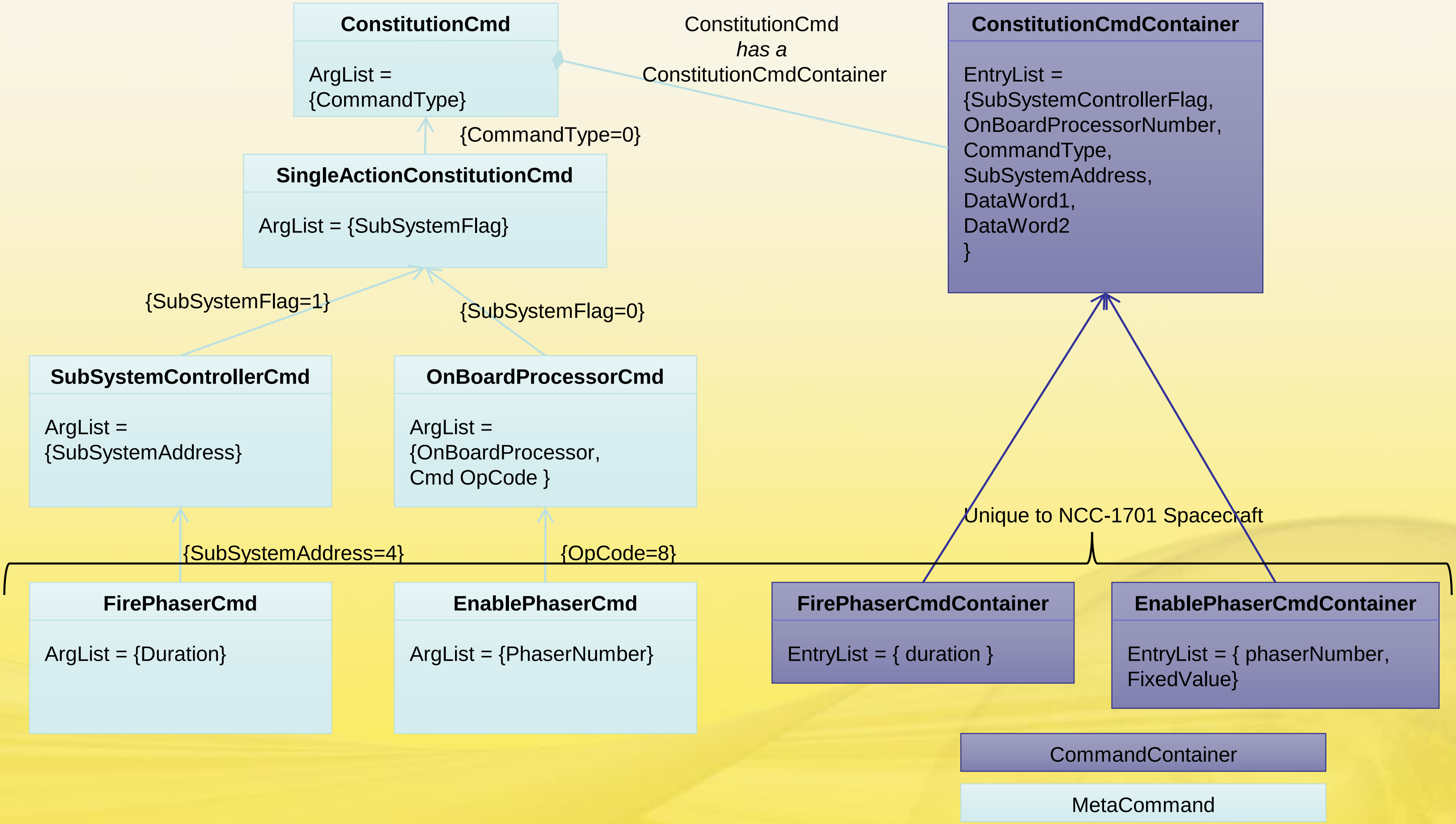
HeaterOff
-Arg1: SubSystemName=ThermalControl
-Arg2: DeviceName=Heater1
-Arg3: State=OFF

Commanding – MetaCommand’s CommandContainer

- MetaCommand/CommandContainer
  - Similar to the other Containers
    - It has a FixedValueEntry to hard code a value
    - It has an ArgumentRefEntry for Arguments
      - (and ArrayArgument and AggrateArgument)
    - It has a BaseContainer but its RestrictionCriteria is optional
    - It has no abstract attribute either

- Use it to build the Packet or Container associated with the MetaCommand, the EntryList can have:
  - ParameterRefEntry and its various forms
  - ArgumentRefEntry, ArgumentArrayRefEntry, ArgumentAggregateRefEntry
  - FixedValueEntry
  - ContainerRefEntry and its various forms
    - Don’t Ref another MetaCommand/Container
    - Instead Ref in CommandContainerSet!

Commanding – Example as UML



ConstitutionCmd

ArgList = {CommandType}

ConstitutionCmd *has a* ConstitutionCmdContainer

ConstitutionCmdContainer

EntryList = {SubSystemControllerFlag, OnBoardProcessorNumber, CommandType, SubSystemAddress, DataWord1, DataWord2 }

{CommandType=0}

SingleActionConstitutionCmd

ArgList = {SubSystemFlag}

{SubSystemFlag=1}

{SubSystemFlag=0}

SubSystemControllerCmd

ArgList = {SubSystemAddress}

OnBoardProcessorCmd

ArgList = {OnBoardProcessor, Cmd OpCode }

Unique to NCC-1701 Spacecraft

{SubSystemAddress=4}

{OpCode=8}

FirePhaserCmd

ArgList = {Duration}

EnablePhaserCmd

ArgList = {PhaserNumber}

FirePhaserCmdContainer

EntryList = { duration }

EnablePhaserCmdContainer

EntryList = { phaserNumber, FixedValue}

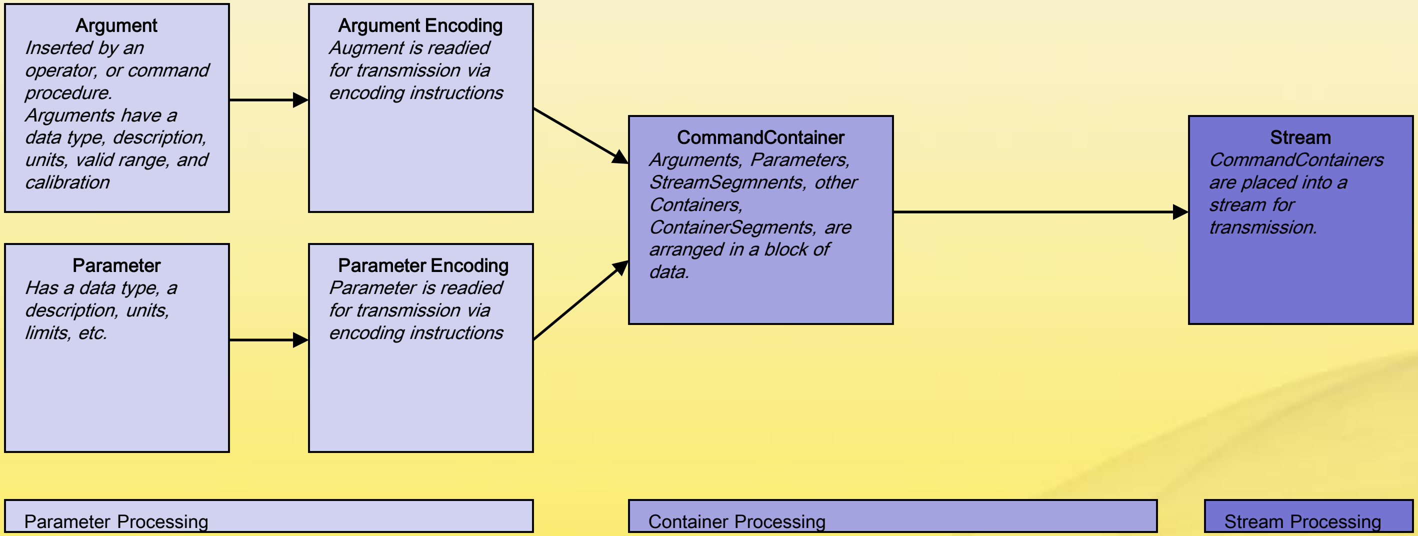
CommandContainer

MetaCommand

29 Feb 2016

Commanding - Processing Flow

Similar to Telemetry processing flow



Argument
Inserted by an operator, or command procedure.
Arguments have a data type, description, units, valid range, and calibration

Argument Encoding
Augment is readied for transmission via encoding instructions

Parameter
Has a data type, a description, units, limits, etc.

Parameter Encoding
Parameter is readied for transmission via encoding instructions

CommandContainer
Arguments, Parameters, StreamSegmnents, other Containers, ContainerSegments, are arranged in a block of data.

Stream
CommandContainers are placed into a stream for transmission.

Parameter Processing

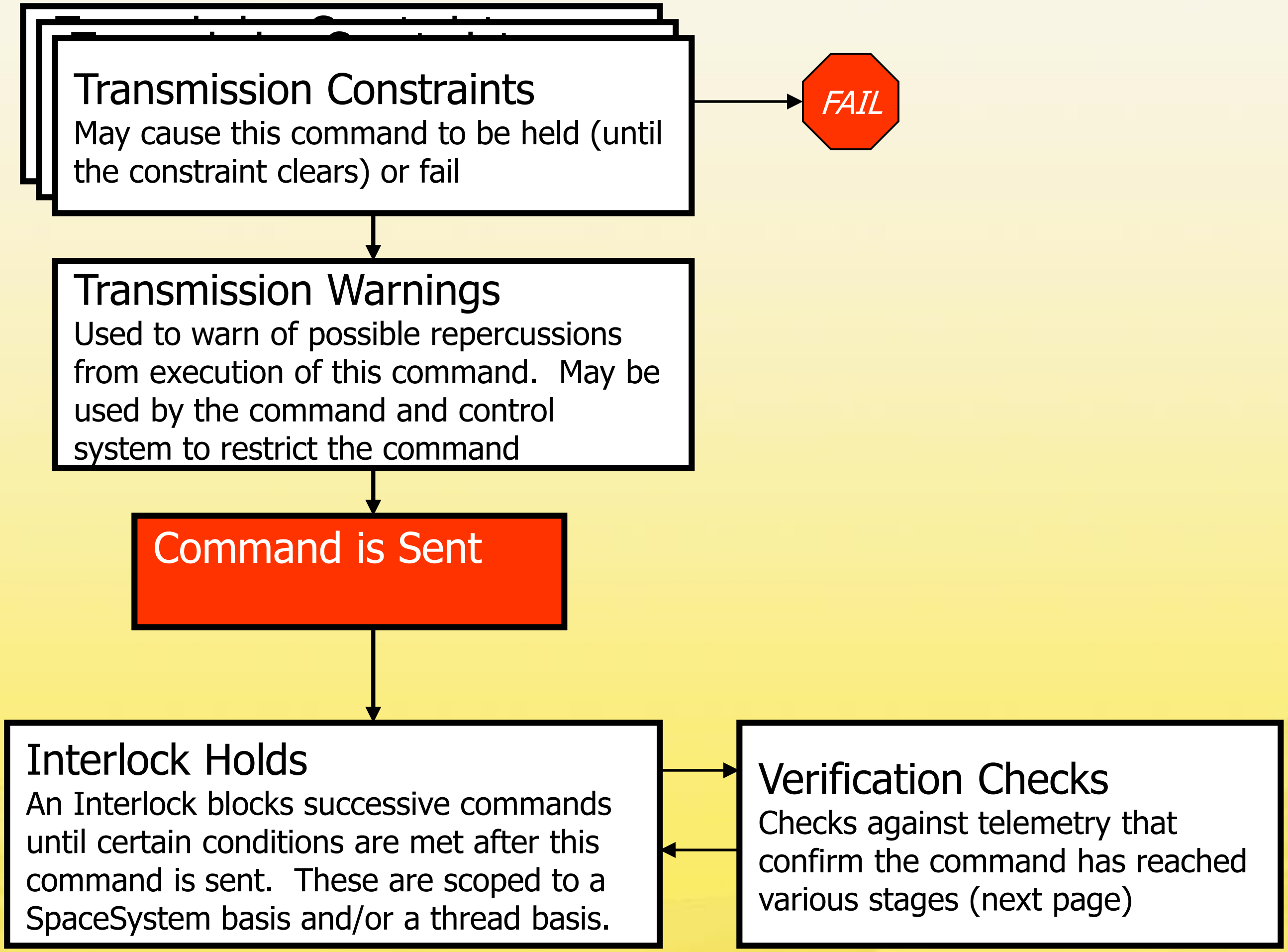
Container Processing

Stream Processing

Commanding – Command Processing

- Other items in MetaCommand are oriented towards command processing:
  - Either procssing associated with sending it
  - Or expected results after it is issued (side effects, etc…)
- These elements are all optional:
  - TransmissionConstraint - <TransmissionConstraintList>
    - Check cmd can be run -- Implied blockage if constraints fail
  - Significance - <DefaultSignificance>, <ContextSignificanceList>
    - Can be used for additional warnings or authentications
  - Interlock
    - Constraints on the next command
  - Verification -- <VerifierSet>
    - Variety of command verification by stages (Command Complete)
  - Set Parameter values– <ParameterToSetList>
    - Side effects after command is verified-- e.g. Command Counter, etc…
  - Suspend Alarms until… <ParameterToSuspendAlarmsSet>
    - Suspend alarms on named parameters while cmd takes effect

Commanding - Execution Sequence



Transmission Constraints
May cause this command to be held (until the constraint clears) or fail

FAIL

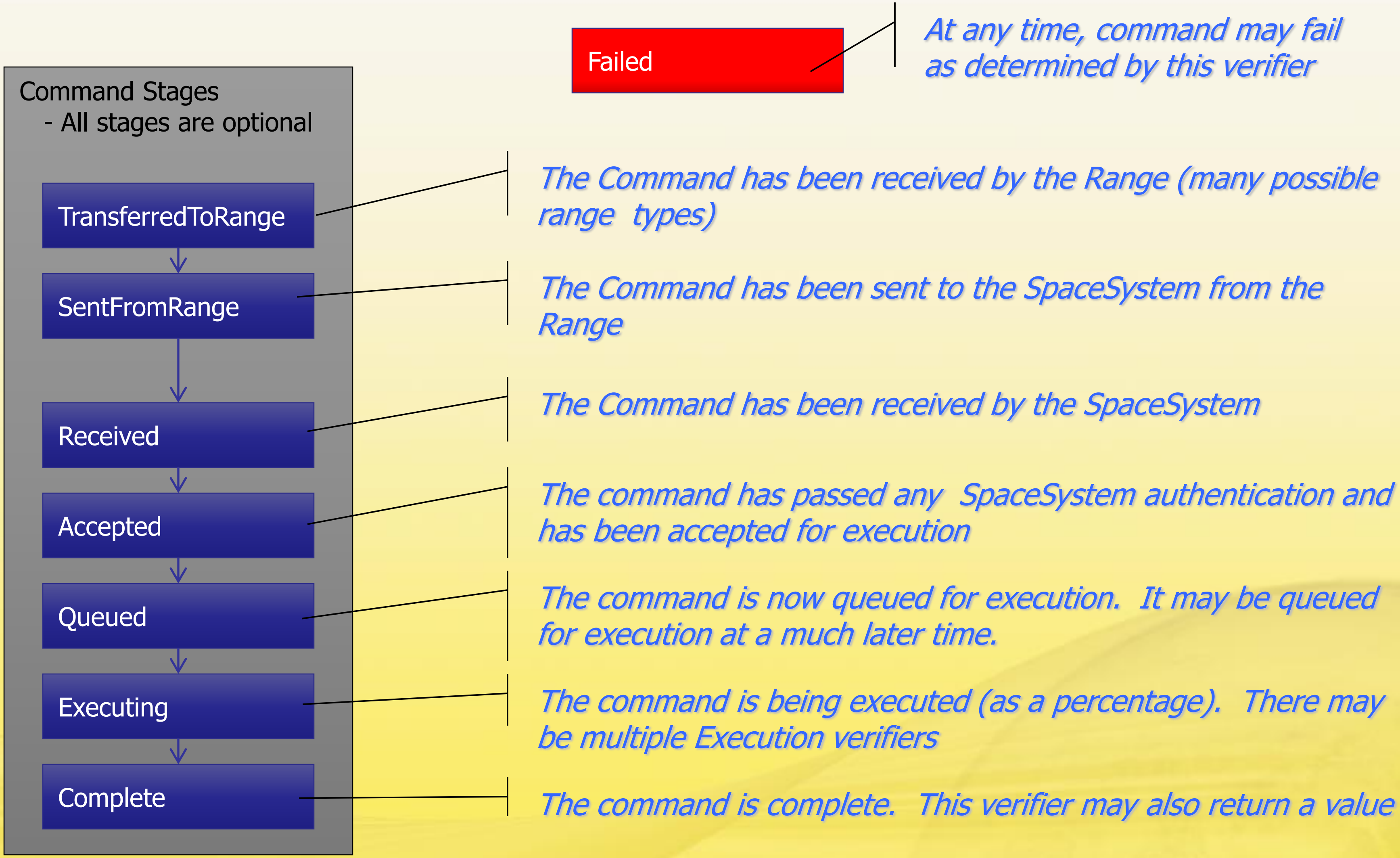
Transmission Warnings
Used to warn of possible repercussions from execution of this command. May be used by the command and control system to restrict the command

Command is Sent

Interlock Holds
An Interlock blocks successive commands until certain conditions are met after this command is sent. These are scoped to a SpaceSystem basis and/or a thread basis.

Verification Checks
Checks against telemetry that confirm the command has reached various stages (next page)

Commanding - Verification Stages



Failed

At any time, command may fail as determined by this verifier

Command Stages
- All stages are optional

TransferredToRange

The Command has been received by the Range (many possible range types)

SentFromRange

The Command has been sent to the SpaceSystem from the Range

Received

The Command has been received by the SpaceSystem

Accepted

The command has passed any SpaceSystem authentication and has been accepted for execution

Queued

The command is now queued for execution. It may be queued for execution at a much later time.

Executing

The command is being executed (as a percentage). There may be multiple Execution verifiers

Complete

The command is complete. This verifier may also return a value

Commanding - Command Significance

- Significance provides some cautionary information about the potential consequence from executing the MetaCommand
  - Ground systems may use this information to request additional confirmation before sending the command
- Default and Context sensitive
- Reason for warning level – simple string explaining significance level
- Consequence Level
  - mirrors alarm levels
    - None
    - Watch
    - Warning
    - Distress
    - Critical
    - Severe

Commanding – MetaCommand Special Handling

- ParametersToSetList
  - A list of Parameters that are to get new values after a Command has passed all Verifications
    - New value can be fixed or derived
- ParametersToSuspendAlarmsOnSet
  - The affects of Commanding can cause alarms to be triggered
  - This set allows some alarms to be suspended for a given period of time after a given verifier is met

Commanding – CommandContainerSet

```
<xtce:SpaceSystem>
  <xtce:TelemetryMetaData>
      <xtce:ParameterTypeSet/>
      <xtce:ParameterSet/>
      <xtce:ContainerSet/>
  </xtce:TelemetryMetaData>
  <xtce:CommandMetaData>
      <xtce:ParameterTypeSet/>
      <xtce:ParameterSet/>
      <xtce:ArgumentTypeSet/>
      <xtce:MetaCommandSet>
            <xtce:MetaCommand>
                 <xtce:ArgumentList/>
                 <xtce:CommandContainer>
            </xtce:MetaCommand>
      </xtce:MetaCommandSet>
      <xtce:CommandContainerSet/>
    </xtce:CommandMetaData>
</xtce:SpaceSystem>
```

CommandContainerSet is NOT the same thing as MetaCommand/CommandContainer

Hello World – In XTCE

29 Feb 2016

TrivalSat Example

- a very simple satellite

Telemetry

0 8 16 24 32

Minor Frame 0	ASM=fa	MFCtr=0	Bat1V	BeaconStatus
Minor Frame 1	ASM=fa	MFCtr=1	BeaconStatus	Bat1V

Bat1V – Battery one voltage, encoded as an 8 bit unsigned integer, MSB first, calibrated to a linear 0 to 32 volt curve

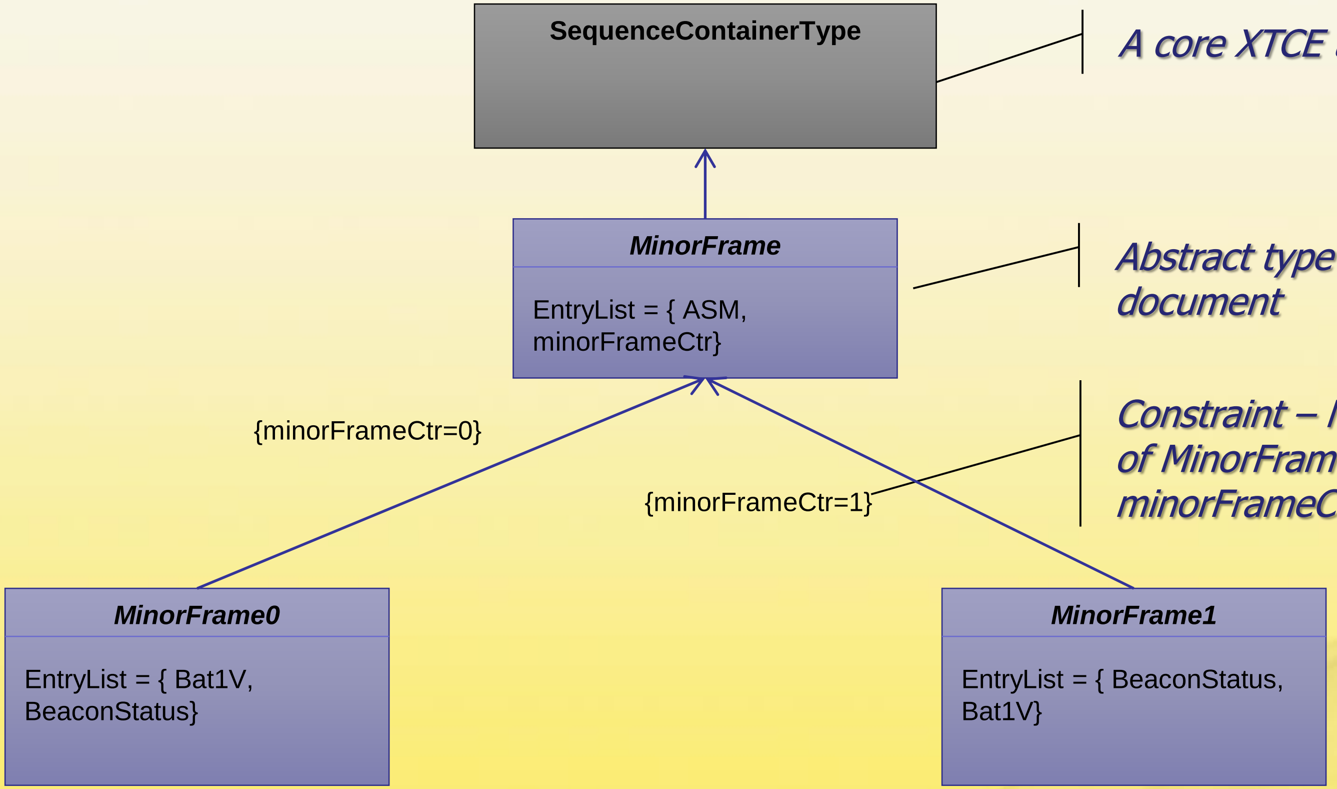
BeaconStatus – Beacon Status, encoded as an 8 bit unsigned integer, MSB first where only the first bit is used, that is treated as a enumerated type where a 1=On and 0=Off.

Command

0 8 16

Command Format	OpCode	Argument(s)
BeaconPwr	OpCode=170	On/Off
BeaconPwrOn	OpCode=170	On/Off=On

TrivialSat – Telemetry UML Taxonomy



SequenceContainerType
MinorFrame
EntryList = { ASM, minorFrameCtr}
{minorFrameCtr=0}
{minorFrameCtr=1}
MinorFrame0
EntryList = { Bat1V, BeaconStatus}
MinorFrame1
EntryList = { BeaconStatus, Bat1V}

A core XTCE type

Abstract type declared in user XML document

Constraint – MinorFrame1 is a type of MinorFrame where the parameter minorFrameCtr = 1

TrivialSat - SpaceSystem

XML Standard

Default namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<SpaceSystem xmlns="http://www.omg.org/space/xtce" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.omg.org/space/xtce SpaceSystemV1.1.xsd" name="TrivialSat">
```

Schema location

SpaceSystem name

```
     <TelemetryMetaData>
          <ParameterTypeSet> <!—details--> </ParameterTypeSet>
          <ParameterSet> <!—details--> </ParameterSet>
          <ContainerSet> <!—details--> </ContainerSet>
     </TelemetryMetaData>
```

Telemetry description goes here

```
     <CommandMetaData>
          <ArgumentTypeSet> <!—details--> </ArgumentTypeSet>
          <ParameterMetaCommandSet> <!—details--> </MetaCommandSet>
          <CommandContainerSet> <!—details--> </CommandContainerSet>
     </CommandMetaData>
</SpaceSystem>
```

Command description
goes here

TrivialSat – ParameterTypeSet

```xml
<ParameterTypeSet>
    <FloatParameterType name="BatteryVoltageType">
        <UnitSet/>
        <IntegerDataEncoding>
            <DefaultCalibrator>
                <PolynomialCalibrator>
                    <Term coefficient="8" exponent="1"/>
                </PolynomialCalibrator>
        </DefaultCalibrator></IntegerDataEncoding>
    </FloatParameterType>
    <EnumeratedParameterType name="BeaconStatusType">
        <UnitSet/>
        <IntegerDataEncoding/>
        <EnumerationList>
            <Enumeration value="128" label="On"/>
            <Enumeration value="0" label="Off"/>
        </EnumerationList>
    </EnumeratedParameterType>
    <BinaryParameterType name="minorFrameCtrType">
        <UnitSet/>
    </BinaryParameterType>
</ParameterTypeSet>
```

Floating Point Parameter

Is encoded as an integer. Default Integer Encoding is unsigned 8 bit MS Bit and byte first

Linear calibration

Enumeration Parameter

Encoded as an 8 bit unsigned measurand

8 Bit Binary Parameter

TrivialSat – ParameterSet

```
<ParameterSet>

    <Parameter name="BatteryVoltage" parameterTypeRef="BatteryVoltageType"/>

    <Parameter name="BeaconStatus" parameterTypeRef="BeaconStatusType"/>

    <Parameter name="minorFrameCtr" parameterTypeRef="minorFrameCtrType"/>

</ParameterSet>
```

Parameter will be of this type

Declare a Parameter with this name

TrivialSat – ContainerSet

```
<ContainerSet>
        <SequenceContainer name="MinorFrame">
                <EntryList>
                        <ParameterRefEntry parameterRef="minorFrameCtr"/>
                </EntryList>
        </SequenceContainer>
        <SequenceContainer name="MinorFrame0">
                <EntryList>
                        <ParameterRefEntry parameterRef="Battery1Voltage"/>
                        <ParameterRefEntry parameterRef="BeaconStatus"/>
                </EntryList>
                <Base containerRef="MinorFrame">
                        <RestrictionCriteria>
                                <Comparison parameterRef="minorFrameCtr" value="0x00"/>
                        </RestrictionCriteria>
                </Base>
        </SequenceContainer>
        <SequenceContainer name="MinorFrame1">
        ….
        </SequenceContainer>
</ContainerSet>
```

Abstract MinorFrame

MinorFrame0

IS A MinorFrame

Where the minorFrameCtr =0

TrivialSat - ArgumentTypes

```
<ArgumentTypeSet>
    <BinaryArgumentType name="opCodeType">
        <UnitSet/>
    </BinaryArgumentType>
    <EnumeratedArgumentType name="onOffType">
        <UnitSet/>
        <EnumerationList>
            <Enumeration value="1" label="on"/>
            <Enumeration value="0" label="off"/>
        </EnumerationList>
    </EnumeratedArgumentType>
</ArgumentTypeSet>
```

TrivialSat – MetaCommand: Top level MetaCommand

```
<MetaCommand name="TrivialSatCmdType" abstract="true">
    <ArgumentList>
        <Argument name="opCode" argumentTypeRef="opCodeType"/>
    </ArgumentList>
    <CommandContainer name="TrivialSatCommandContainerType">
        <EntryList>
            <ArgumentRefEntry argumentRef="opCode"/>
        </EntryList>
    </CommandContainer>
</MetaCommand>
```

Abstract TrivalSatCmd – only has an opCode

TrivialSat – MetaCommand: BeaconPwr

```
<MetaCommand name="BeaconPwr">
    <LongDescription>Turn Beacon Power on or off</LongDescription>
    <BaseMetaCommand metaCommandRef="TrivialSatCmdType">
        <ArgumentAssignmentList>
            <ArgumentAssignment argumentName="opCode" argumentValue="170"/>
        </ArgumentAssignmentList>
    </BaseMetaCommand>
    <ArgumentList>
        <Argument name="powerValue" argumentTypeRef="onOffType"/>
    </ArgumentList>
    <CommandContainer name="BeaconPwr">
        <EntryList>
            <ArgumentRefEntry argumentRef="powerValue"/>
        </EntryList>
        <BaseContainer containerRef="TrivialSatCmdContianerType">
        </BaseContainer>
    </CommandContainer>
</MetaCommand>
```

Value of opCode is fixed for this MetaCommand type

New argument introduced

MetaCommand container – the powerValue argument is added to the container

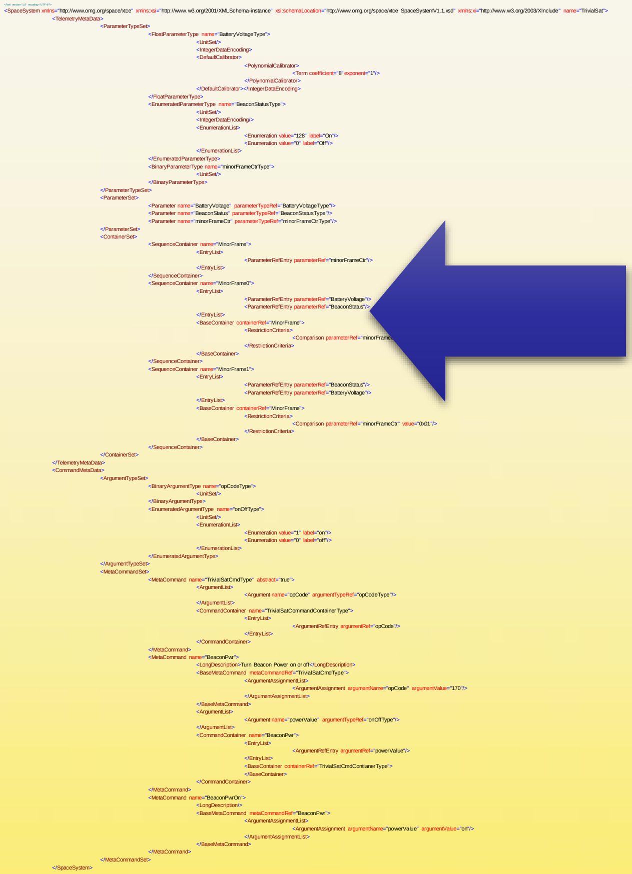
TrivialSat – MetaCommand: BeconPwrOn

Type of BeconPwr Command

```
<MetaCommand name="BeaconPwrOn">
        <LongDescription/>
        <BaseMetaCommand metaCommandRef="BeaconPwr">
                <ArgumentAssignmentList>
                        <ArgumentAssignment argumentName="powerValue" argumentValue="on"/>
                </ArgumentAssignmentList>
        </BaseMetaCommand>
</MetaCommand>
```

Argument is fixed to 'on'

TrivialSat – Complete XML



```xml
<?xml version="1.0" encoding="UTF-8"?>
<SpaceSystem xmlns="http://www.omg.org/space/xtce" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.omg.org/space/xtce SpaceSystemV1.1.xsd" xmlns:xi="http://www.w3.org/2003/XInclude" name="TrivialSat">
    <TelemetryMetaData>
        <ParameterTypeSet>
            <FloatParameterType name="BatteryVoltageType">
                <UnitSet/>
                <IntegerDataEncoding>
                <DefaultCalibrator>
                    <PolynomialCalibrator>
                        <Term coefficient="8" exponent="1"/>
                    </PolynomialCalibrator>
                </DefaultCalibrator></IntegerDataEncoding>
            </FloatParameterType>
            <EnumeratedParameterType name="BeaconStatusType">
                <UnitSet/>
                <IntegerDataEncoding/>
                <EnumerationList>
                    <Enumeration value="128" label="On"/>
                    <Enumeration value="0" label="Off"/>
                </EnumerationList>
            </EnumeratedParameterType>
            <BinaryParameterType name="minorFrameCtrType">
                <UnitSet/>
            </BinaryParameterType>
        </ParameterTypeSet>
        <ParameterSet>
            <Parameter name="BatteryVoltage" parameterTypeRef="BatteryVoltageType"/>
            <Parameter name="BeaconStatus" parameterTypeRef="BeaconStatusType"/>
            <Parameter name="minorFrameCtr" parameterTypeRef="minorFrameCtrType"/>
        </ParameterSet>
        <ContainerSet>
            <SequenceContainer name="MinorFrame">
                <EntryList>
                    <ParameterRefEntry parameterRef="minorFrameCtr"/>
                </EntryList>
            </SequenceContainer>
            <SequenceContainer name="MinorFrame0">
                <EntryList>
                    <ParameterRefEntry parameterRef="BatteryVoltage"/>
                    <ParameterRefEntry parameterRef="BeaconStatus"/>
                </EntryList>
                <BaseContainer containerRef="MinorFrame">
                    <RestrictionCriteria>
                        <Comparison parameterRef="minorFrame
                    </RestrictionCriteria>
                </BaseContainer>
            </SequenceContainer>
            <SequenceContainer name="MinorFrame1">
                <EntryList>
                    <ParameterRefEntry parameterRef="BeaconStatus"/>
                    <ParameterRefEntry parameterRef="BatteryVoltage"/>
                </EntryList>
                <BaseContainer containerRef="MinorFrame">
                    <RestrictionCriteria>
                        <Comparison parameterRef="minorFrameCtr" value="0x01"/>
                    </RestrictionCriteria>
                </BaseContainer>
            </SequenceContainer>
        </ContainerSet>
    </TelemetryMetaData>
    <CommandMetaData>
        <ArgumentTypeSet>
            <BinaryArgumentType name="opCodeType">
                <UnitSet/>
            </BinaryArgumentType>
            <EnumeratedArgumentType name="onOffType">
                <UnitSet/>
                <EnumerationList>
                    <Enumeration value="1" label="on"/>
                    <Enumeration value="0" label="off"/>
                </EnumerationList>
            </EnumeratedArgumentType>
        </ArgumentTypeSet>
        <MetaCommandSet>
            <MetaCommand name="TrivialSatCmdType" abstract="true">
                <ArgumentList>
                    <Argument name="opCode" argumentTypeRef="opCodeType"/>
                </ArgumentList>
                <CommandContainer name="TrivialSatCommandContainerType">
                    <EntryList>
                        <ArgumentRefEntry argumentRef="opCode"/>
                    </EntryList>
                </CommandContainer>
            </MetaCommand>
            <MetaCommand name="BeaconPwr">
                <LongDescription>Turn Beacon Power on or off</LongDescription>
                <BaseMetaCommand metaCommandRef="TrivialSatCmdType">
                    <ArgumentAssignmentList>
                        <ArgumentAssignment argumentName="opCode" argumentValue="170"/>
                    </ArgumentAssignmentList>
                </BaseMetaCommand>
                <ArgumentList>
                    <Argument name="powerValue" argumentTypeRef="onOffType"/>
                </ArgumentList>
                <CommandContainer name="BeaconPwr">
                    <EntryList>
                        <ArgumentRefEntry argumentRef="powerValue"/>
                    </EntryList>
                    <BaseContainer containerRef="TrivialSatCmdContianerType">
                    </BaseContainer>
                </CommandContainer>
            </MetaCommand>
            <MetaCommand name="BeaconPwrOn">
                <LongDescription/>
                <BaseMetaCommand metaCommandRef="BeaconPwr">
                    <ArgumentAssignmentList>
                        <ArgumentAssignment argumentName="powerValue" argumentValue="on"/>
                    </ArgumentAssignmentList>
                </BaseMetaCommand>
            </MetaCommand>
        </MetaCommandSet>
    </SpaceSystem>
```

The entire XML document for TrivialSat

Nuances

29 Feb 2016

Nuances – Integer Values

- May be specified as a decimal number: e.g., 1234
- May be specified as a hexadecimal number: e.g., 0xFA or 0Xfa
- May be specified as an octal number: e.g., 0o123 or 0O123
- May be specified as a binary number: e.g., 0b0110 or 0B0110
- If the size is smaller than the number given, the number is truncated from the most significant side

Nuances – Units

- Units in XTCE were a compromise between the very complex and overly simple.
- All Data Types (Parameters and Command Arguments) have a mandatory UnitSet – the UnitSet may be empty.
- Unit has a factor and a power. Multiple Units are multiplied.

```xml
<FloatParameter name="DarkEnergySensor">
    <UnitSet>
        <Unit>Joules</Unit>
        <Unit power="-4">m</Unit>
    </UnitSet>
    <IntegerDataEncoding sizeInBits="7" encoding="unsigned"/>
</FloatParameter>
```

$$= \frac{Joules}{m^4}$$

Nuances - ParameterInstanceRefs

- Used when it’s necessary to refer to a specific instance of a Parameter.
- Used when the value of the Parameter is used for a calculation or a reference
- The reference is either positive for forward or negative for backward in time, 0 is the current value or the very first instance of the Parameter within a container of concern.

Nuances – Lessor Elements

- Lesser used major elements include:
  - TelemetryMetaData/MessageSet
  - StreamSet
  - AlgorithmSet
  - ServiceSet
  - AliasSet
  - AncillaryDataSet
  - PhysicalAddressSet

Nuances - MessageSet

- Use is not recommended
- Essentially an alternative way to build a Packet or Minor Frame…
  - A purely “HAS A” relationship to Containers
  - Retained from an earlier version of XTCE

Nuances - StreamSet – Streams

- A set of elements to describe aspects of “bit streams”
  - – Can be used for portions of frame description, etc…
- May be included in containers directly
  - – For embedded steams – i.e., “sub-stream” in a container
- CCSDS – not quite enough elements to FULLY describe the “CCSDS stack” from the frame-sync to packets
  - – Lacks RS encoding info for example (although one could use the ‘CustomStream’)
  - – Generally focus on packet descriptions and leave the FEP for others
- FixedFrameStreams or VariableFrameStreams reference a top level abstract container
    - SequenceContainer identification techniques will determine the specific container
- Or links to a service
  - – Services are abstractions, perhaps a service has certain streams within it…

Nuances - Streams Types

- **PCMStreamType** - A PCM Stream Type is the high level definition for all Pulse Code Modulated (PCM) (i.e., binary) streams. Attributes include the rate, the PCM type, and a flag for inverted streams.
- **FrameStreamType** - The top level type definition for all data streams that are frame based.
- **FixedFrameStreamType** - For streams that contain a series of frames with a fixed frame length where the frames are found by looking for a marker in the data. Attributes include the marker.
- **VariableFrameStreamType** - For streams that contain a series of frames with a variable frame length where the frames are found by looking for a series of one's or zero's (usually one's). The series is called the flag and is made impossible to appear in the data by zero or one bit insertion.
- **CustomStreamType** - A stream type where some level of custom processing (e.g. convolutional, encryption, compression) is performed. Has a reference to external algorithms for encoding and decoding algorithms.

Nuances – AlgorithmSet

- Used to describe algorithms during tlm/cmd processing
  - Many options – from actually including code text, to simply the name of a function…
- Two forms:
  - Custom: functions, methods, procedures…
  - Math: equations using postfix notation…
- Usually define a ‘trigger’ that indicates when the algorithm should be invoked

Nuances - Algorithms

- XTCE does not have it’s own algorithm language, but the database format must still refer to external algorithms from time to time
- The basic algorithm type simply refers to some external algorithm
  - External algorithm may be a script, a shared library name, a Java Object file, etc.
  - Multiple references may be included so that a single XTCE file may be used across multiple ground systems.

Nuances - Algorithm Inputs and Outputs

- Some Algorithms allow a set of inputs
  - Inputs are named ParameterInstanceRefs or
  - Named Constants
- Some Algorithms also allow a set of outputs
  - Outputs are named ParametersRefs

Algorithm Triggers

- Triggers are used to initiate the processing of some algorithms. A trigger may be based on an update of a Parameter or on a time (periodic) basis.
- Triggers may also have a rate that limits their firing to a 1/rate basis.

Nuances - Services

- An abstraction – groups “logically like” Containers
- For example:
  - Suppose your mission splits its functionality across multiple CCSDS Virtual Channels
  - Each Channel may represent a “service”
  - A Service element could be defined and the list of Containers for each APID in the service (VCID) added
    - Options:
      - point to the “super container” per channel
      - Point to each “final” packet container
      - Point to all the containers making up packets on those channels…
      - Etc…?
  - Totally user defined…

Nuances – Aliases Example

- Aliases –all major Elements may have an unlimited number of aliases. Aliases may be used for ground IDs, Onboard IDs, mnemonics etc.
- Each Alias has a nameSpace
- Example:

```
<AliasSet>
    <Alias nameSpace=“Mnemonic" alias=“BAT1I"/>
    <Alias nameSpace="OnBoardID" alias="020660"/>
</AliasSet>
```

Nuances – Ancillary Data Example

- Use for any other data associated with the objects.
- May be used to include administrative data (e.g., version, CM or tags) or potentially any MIME type.
- Data may be included or given as an href.
- Example:

```
<AncillaryDataSet>
        <AncillaryData name="CommandIcon" mimeType="image/jpeg" href="http://somewhere.com/batt.jpeg"/>
        <AncillaryData name="FactoryVersion">8.5a</AncillaryData>
        <AncillaryData name="OperationsCenterVersion">2.2.1</AncillaryData>
        <AncillaryData name="CmdCpltSnd" mimeType="audio/mpeg" href="http://muzik.com/tada.mpeg"/>
</AncillaryDataSet>
```

Nuances – Physical Address Example

- Physical Address’s are most often used by the Spacecraft factory
- Contains a sourceName, a sourceAddress, and a SubAddress (arbitrarily deep)
- There many be any number of Physical Address’s

```
<PhysicalAddressSet>
    <PhysicalAddress sourceName="TransducerIP" sourceAddress="143.57.15.01">
        <SubAddress sourceAddress="TransducerCard" sourceName="8">
            <SubAddress sourceName="portNumber" sourceAddress="14"/>
        </SubAddress>
    </PhysicalAddress>
</PhysicalAddressSet>
```

XUSP Tutorial

Four-letter acronyms that make a difference in ground system interoperability

GSAW MMXVI

Prepared By The OMG Space Domain Task Force

And

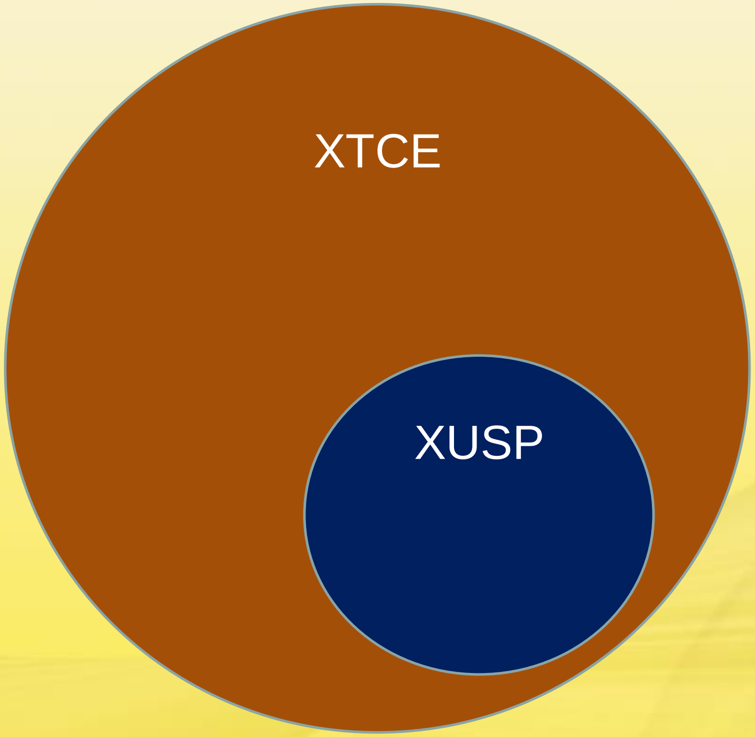
HARRIS®

Published by The Aerospace Corporation with permission.

OBJECT MANAGEMENT GROUP

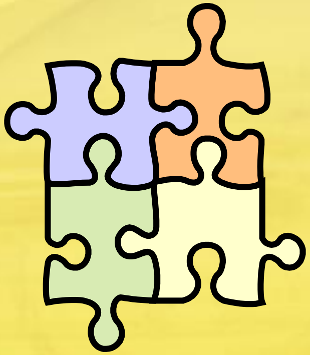
XUSP

- XTCE US government Satellite conformance Profile
  - XUSP is a proper subset of the XTCE specification. An XUSP 1.0 document must be XTCE 1.1-compliant.



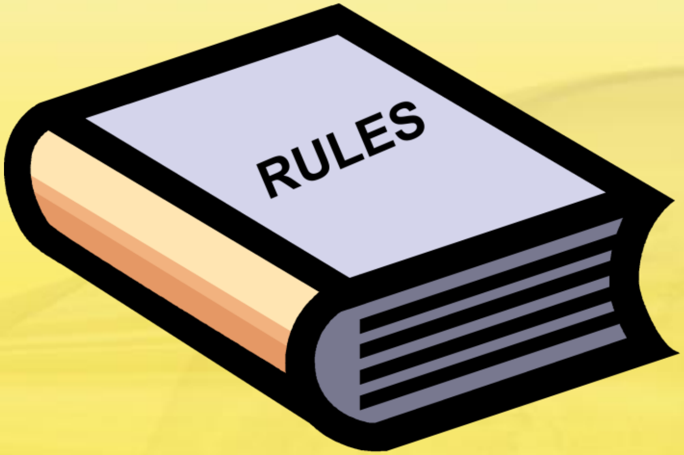
Why Does XTCE Need a Profile?

- XTCE was designed primarily as an exchange format, so very few limitations were put into the schema
  - Unlimited parameter/argument name lengths
  - Unlimited polynomial coefficients
  - Unlimited nesting of SpaceSystems
- Real ground systems have limitations
- A profile allow deletions, additions, and specializations to a specification to make it better fit a specific purpose, in this case, transfer of command and telemetry definitions for a US government satellite with CCSDS space link packet structures between participating organizations
- Reduce the cost of compliant implementations



Structure of the XUSP Specification

- An XML template for compliant documents supplied as a machine-readable file.
- A rules table written in Xpath and supplied as a machine-readable file.
- Additional rules that cannot be easily implemented as Xpath are described in the specification.
- Guidelines for extending the telemetry and command metadata



The XUSP Template

- The XUSP template is to be used as a base for all XUSP documents.
  - All elements in the template must be present
  - SpaceSystem unique containers (command and telemetry) all derive from the template elements
  - Seven attributes in the template should be modified to describe the specific mission:

/SpaceSystem@name	(Mission name)
/SpaceSystem@shortDescription	(Mission description)
/SpaceSystem/AliasSet/Alias@alias	(numeric Spacecraft ID)
/SpaceSystem/Header@version	(XTCE document version)
/SpaceSystem/Header@validationStatus	
/SpaceSystem/Header@date	
/SpaceSystem/Header@classification	

- http://www.omg.org/spec/XUSP/20140801/XUSP.xml

Xpath Validation

- The rules table is a comma-separated-values file with a row for each XTCE Element/Attribute (2037 rows)

- There is a supported column that contains "Supported" if the element/attribute is allowed in XUSP, or an "X" if the element/attribute is not allowed.

- Some rows contain an Xpath 2.0 expression to either limit a supported element/attribute or to verify that an element/attribute is not present.
  - 1480 supported elements have a limitation
  - 460 unsupported elements have count()=0 validation
  - e.g. count(//xtce:SpaceSystem)=1

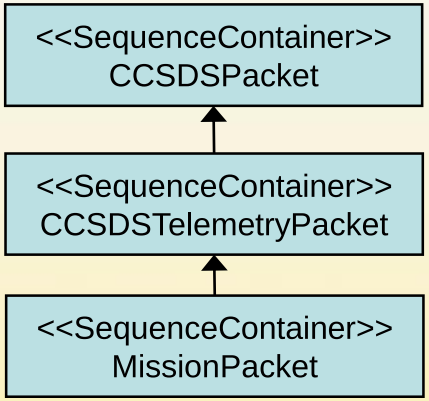
- Some of the limitations:
  - Only one SpaceSystem (no hierarchical nesting of subsystems)
  - Spacsystem, Parameter, Argument, and Command names must be 1-64 characters
  - Only three alarm levels
  - 10th order polynomials

Declaring the VCID's for a Packet

```
<xtce:SequenceContainer name=“MyPacket">
  <xtce:AncillaryDataSet>
    <xtce:AncillaryData name="VCID">0, 20, 8-12</xtce:AncillaryData>
  </xtce:AncillaryDataSet>
  <xtce:EntryList>
     <xtce:ParameterRefEntry parameterRef="TimeStamp"/>
     <xtce:ParameterRefEntry parameterRef="NumImagers"/>
  </xtce:EntryList>
</xtce:SequenceContainer>
```

Extending the Template for Telemetry

- All mission telemetry packets will inherit their structure from the CCSDSTelemetryPacket SequenceContainer in the template.
  - Template defines the CCSDS packet header
  - RestrictionCriteria includes APID and up to 3 other fields



```
<xtce:SequenceContainer name="EPSStatusPacket">
   <xtce:LongDescription>Electrical Power Subsystem normal status.</xtce:LongDescription>
   <xtce:EntryList>
      <xtce:ParameterRefEntry parameterRef="BusAstate">
      <xtce:ParameterRefEntry parameterRef="BusBstate">
  <xtce:EntryList/>
   <xtce:BaseContainer containerRef="CCSDSTelemetryPacket">
      <xtce:RestrictionCriteria><xtce:ComparisonList>
         <xtce:Comparison value="400" parameterRef="CCSDSAPID"/>
   </xtce:ComparisonList></xtce:RestrictionCriteria>
   </xtce:BaseContainer>
</xtce:SequenceContainer>
```

Extending the Template for Commands

```
<xtce:MetaCommand abstract="true" name="SetRelays">
  <xtce:LongDescription>Set relay base command</xtce:LongDescription>
  <xtce:BaseMetaCommand metaCommandRef="CCSDSCommand"/>
  <xtce:ArgumentList>
    <xtce:Argument name="Relay" argumentTypeRef="RelayType"/>
    <xtce:Argument name="RelayState” argumentTypeRef="RelayStateType"/>
 </xtce:ArgumentList>
 <xtce:CommandContainer name="SetRelayPacket"
        shortDescription="Turn a relay off or on, or unchanged">
    <xtce:AncillaryDataSet>
       <xtce:AncillaryData name="VCID">0</xtce:AncillaryData>
    </xtce:AncillaryDataSet>
    <xtce:EntryList>
       <xtce:ArgumentRefEntry argumentRef="Relay"/>
       <xtce:ArgumentRefEntry argumentRef="RelayState"/>
    </xtce:EntryList>
    <xtce:BaseContainer containerRef="CCSDSCommandPacket">
       <xtce:RestrictionCriteria>
         <xtce:Comparison value="1" parameterRef="CCSDSAPID"/>
```

Command Significance

- Command significance can be used in XUSP to control access to command transmission. A DefaultSignificance can be provided or a ContextSignificanceList can be used to change transmissibility across the mission contexts.
  - “normal” or unspecified = no restrictions
  - “critical” = requires confirmation before transmission
  - “severe” = transmission not allowed

```
<xtce:MetaCommand name=“ArmOrdnanceSet">
   <xtce:LongDescription>Enable ordnance firing circuit</xtce:LongDescription>
   <xtce:BaseMetaCommand metaCommandRef=“OrdnanceCommand"/>
   <xtce:ArgumentList>
      <xtce:Argument name=“OrdnanceSet" argumentTypeRef=“OrdnanceSetType"/>
   </xtce:ArgumentList>
   <xtce:DefaultSignificance consequenceLevel=“critical"/>
</xtce:MetaCommand>
```

Future of XUSP

- XUSP reflects a Community of Practice. XUSP will evolve over time as the specification that it is based on (XTCE 1.1) evolves and the tools and products used by its members change.
  - There was a desire by some FTF members to make an alternate schema file available that incorporates the XUSP Xpath rules into the schema definition to allow faster validation of an XUSP document. This could not be completed and verified for finalization, but may be incorporated as a machine-readable file in a future revision.
  - The evolution of XUSP will be driven by the membership of the Revision Task Force (RTF) chartered by the OMG.

For More Information

- http://www.omg.org/space/ - XUSP specification, other SDTF specs
- bkizzort@harris.com - OMG SDTF co-chair
- Only the beta spec is currently available online, but the final 1.0 rules table is available at:
  - http://www.omg.org/spec/XUSP/20140801/XUSP.xml

GEMS Tutorial

Four-letter acronyms that make a difference in ground system interoperability

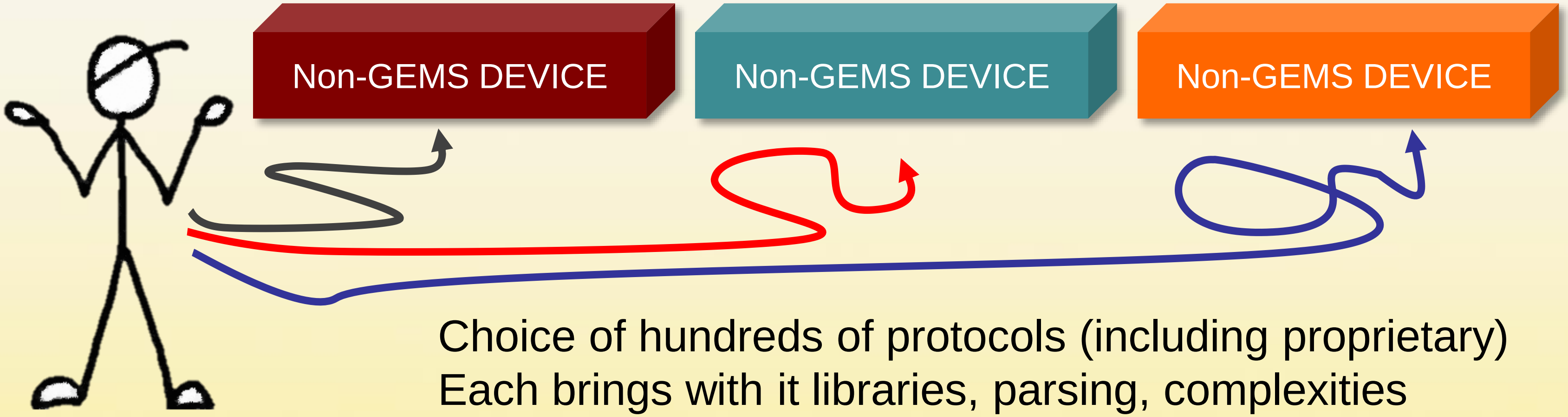
GSAW MMXVI

Prepared By The OMG Space Domain Task Force
And

AMERGINT
TECHNOLOGIES

Published by The Aerospace Corporation with permission.

GEMS In A Nutshell

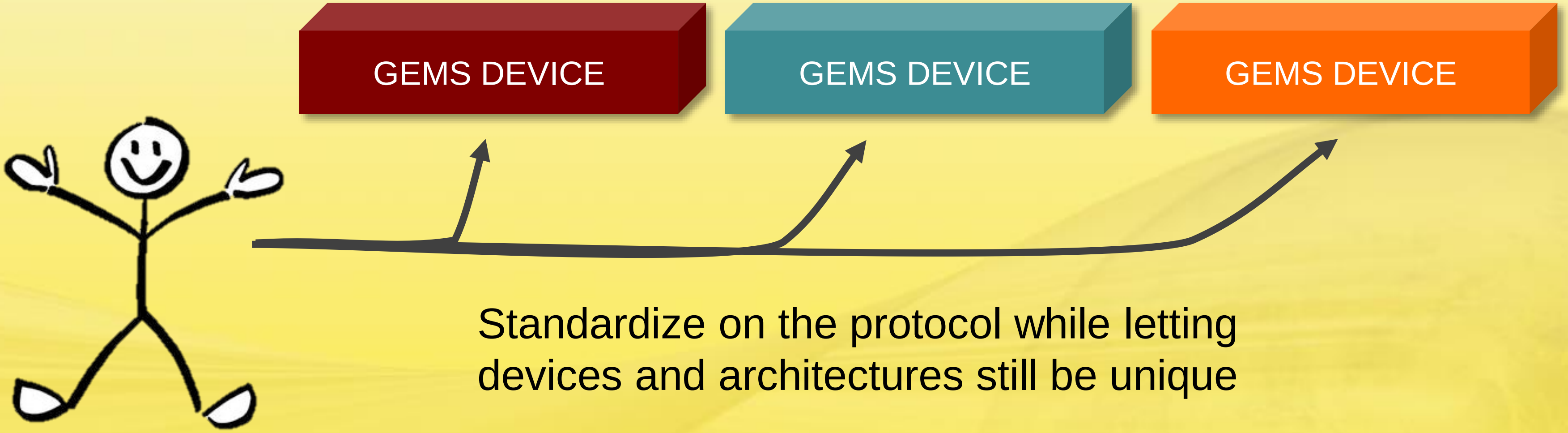


Non-GEMS DEVICE

Non-GEMS DEVICE

Non-GEMS DEVICE

Choice of hundreds of protocols (including proprietary)
Each brings with it libraries, parsing, complexities



GEMS DEVICE

GEMS DEVICE

GEMS DEVICE

Standardize on the protocol while letting devices and architectures still be unique

What Is GEMS?

- Ground Equipment Monitoring Service
  - GEMS is meant to be a simple standard that solves a simple problem – Device Control & Status
  - GEMS defines the protocol structure
    - Does not try to define common parameters

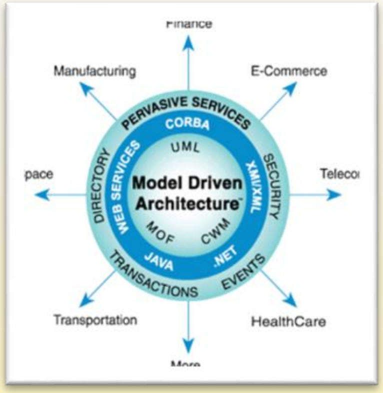
- Why Not Use An Existing Standard?
  - SNMP
    - Neither simple or well suited for what we do
  - CORBA, JMS, SOAP, etc…
    - Forces technology in to your architecture

http://space.omg.org/ → GEMS Wiki

http://www.omgwiki.org/space/doku.php?id=gems

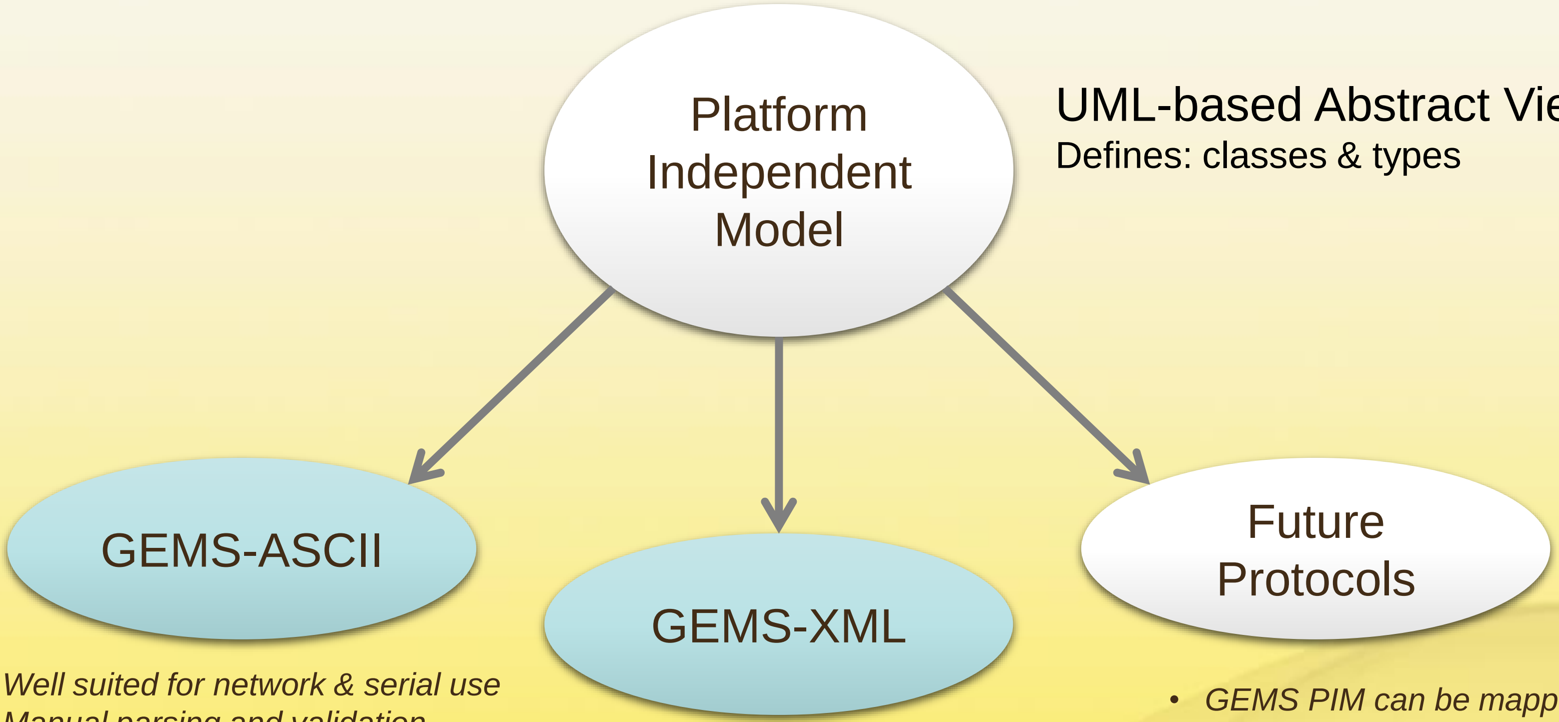
GEMS Guiding Principles

- Keep It Simple!
  - Poll The Device
    - No need for publish/subscribe or other similar features
  - Must be human readable
    - Easy to implement & troubleshoot
  - Adapt to multiple formats
    - Not a one-size fits all standard
  - Adapt to multiple transport mechanisms
    - Not just a network-based standard
  - Scalable
    - Not all systems are small



- Use OMG’s Model Driven Architecture (MDA)
  - Provides an abstract model independent of the specific protocol & transport mechanism

GEMS Use Of MDA



UML-based Abstract View

Defines: classes & types

- *Well suited for network & serial use*
- *Manual parsing and validation*
- *Relatively low bandwidth*

- *Well suited for network & file use*
- *Libraries for parsing and validation*
- *Relatively high bandwidth*

- *GEMS PIM can be mapped to multiple protocols*
- *Ex: SNMP, CORBA, GMSEC*

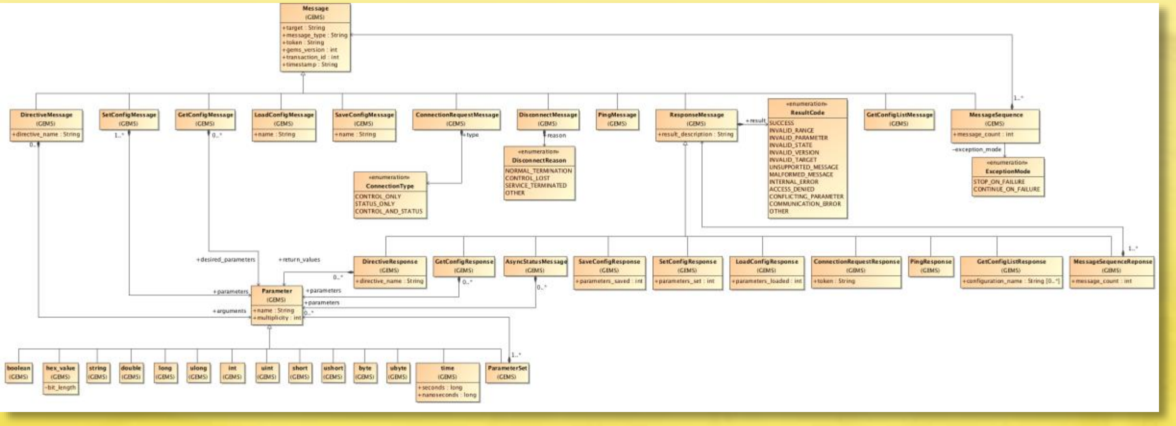
GEMS Version History (Significant Changes)

Type	Description
Version 1.1	Adjusted the hex_value representation in GEMS-ASCII Changed the escape characters Added utime (UTC)
Version 1.2	Added the UnknownResponse message type Added Ping Messages Added an example Device Definition file Added HTTP header for XML Changed the GEMS-ASCII versioning
Version 1.3	Increased the message length for GEMS-ASCII Added GEMS Authentication
Version 1.4	Added Asynchronous Status Message and fixed some formatting

Understanding The GEMS PIM

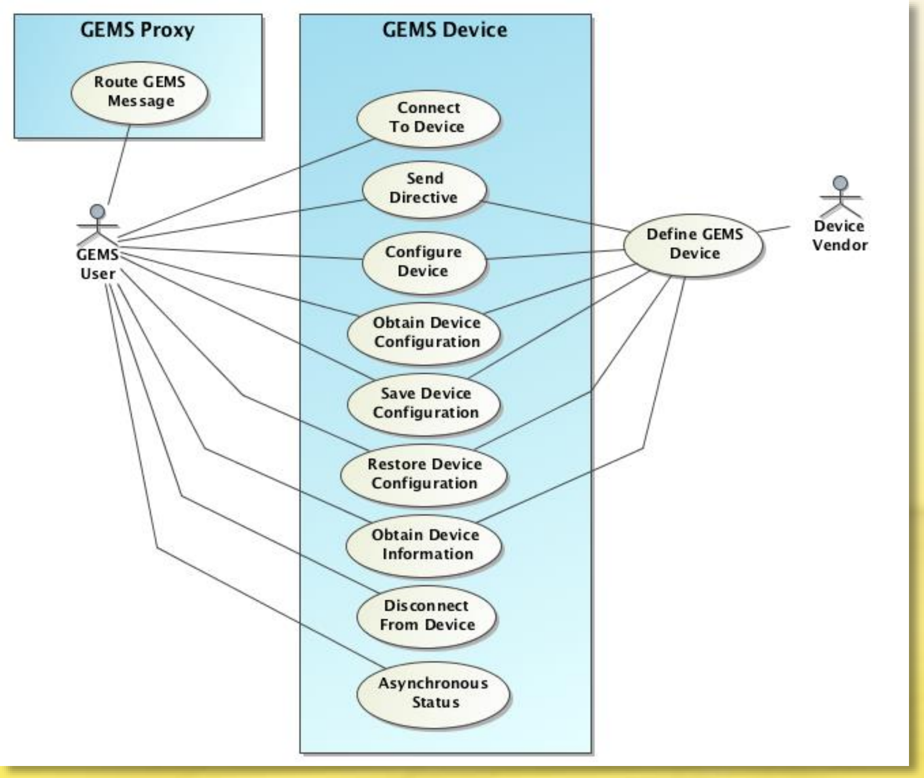
- What Is It?
  - Defines the basic message structure and interaction between a user and a GEMS device

- Includes:
  - Behaviors
  - Parameter Types
  - Message Types
  - Other Features
    - Config Files
    - GEMS device definition files
    - Security
    - Authentication



GEMS Use Cases

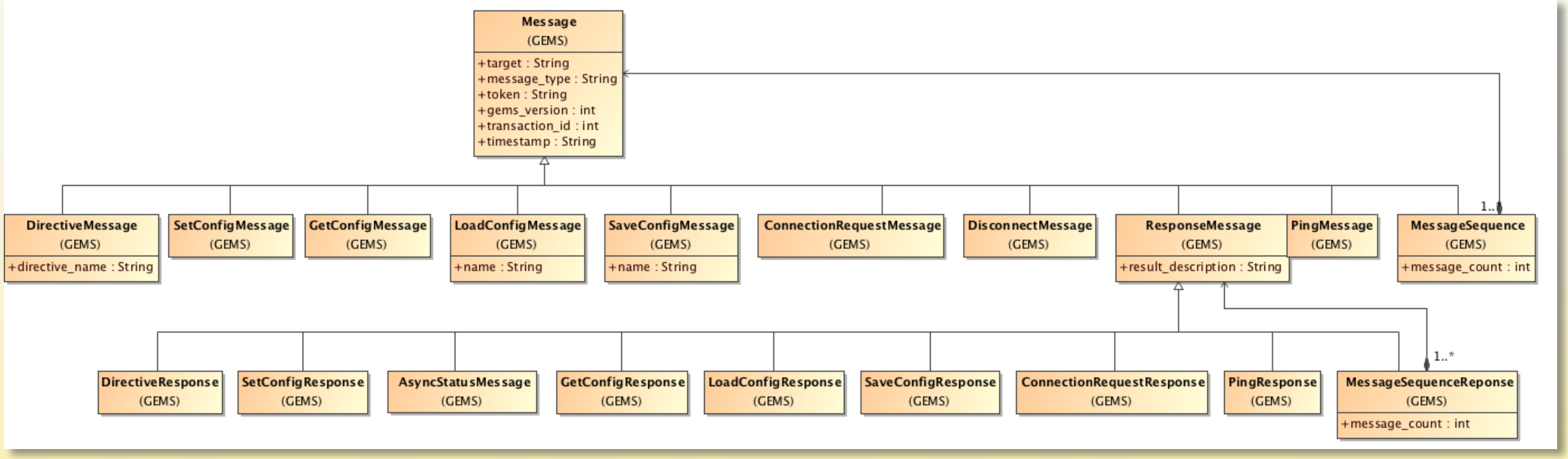
- Basic GEMS Concepts
  - Connecting to a device
  - Configuring a device
  - Getting the configuration
  - Invoking a directive
  - Saving a configuration
  - Restoring a configuration
  - Disconnecting from a device
  - Proxy Messages
  - Discovering device parameters
  - Asynchronous Status



GEMS Parameter Types

Type	Description
boolean	Represents a true/false value
byte	Signed 8 bit byte or octet
ubyte	Unsigned 8 bit byte or octet
long	8 byte integer
ulong	Unsigned 8 byte integer
int	4 byte integer
uint	Unsigned 4 byte integer
short	2 byte integer
ushort	Unsigned 2 byte integer
double	Double precision floating point number
string	Free text ASCII string of characters
hex_value	ASCII representation of a hexadecimal value
time	Represents the number of seconds and nanoseconds since midnight of January 1, 1970
utime	Time value in Coordinated Universal Time (UTC). Allows for the representation of leap seconds.

GEMS Messages



Message
(GEMS)
+target : String
+message_type : String
+token : String
+gems_version : int
+transaction_id : int
+timestamp : String

DirectiveMessage
(GEMS)
+directive_name : String

SetConfigMessage
(GEMS)

GetConfigMessage
(GEMS)

LoadConfigMessage
(GEMS)
+name : String

SaveConfigMessage
(GEMS)
+name : String

ConnectionRequestMessage
(GEMS)

DisconnectMessage
(GEMS)

ResponseMessage
(GEMS)
+result_description : String

PingMessage
(GEMS)

1..*

MessageSequence
(GEMS)
+message_count : int

DirectiveResponse
(GEMS)

SetConfigResponse
(GEMS)

AsyncStatusMessage
(GEMS)

GetConfigResponse
(GEMS)

LoadConfigResponse
(GEMS)

SaveConfigResponse
(GEMS)

ConnectionRequestResponse
(GEMS)

PingResponse
(GEMS)

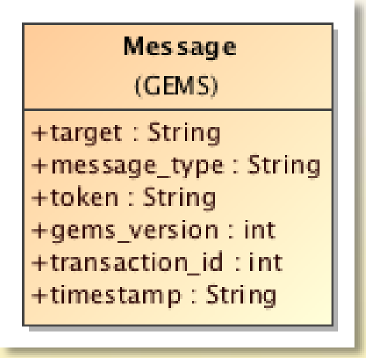
1..*

MessageSequenceReponse
(GEMS)
+message_count : int

Request Message	Description	Response Message
DirectiveMessage	Used to invoke an action	DirectiveResponse
SetConfigMessage	Used to set the configuration of parameters	SetConfigRresponse
GetConfigMessage	Used to obtain the current configuration and status	GetConfigResponse
LoadConfigMessage	Used to load a saved configuration	LoadConfigResponse
SaveConfigMessage	Used to save the current configuration	SaveConfigResponse
ConnectionRequestMessage	Used to connect to a GEMS Device	ConnectionRequestResponse
PingMessage	Used to determine if a GEMS Device is still active	PingResponse
MessageSequence	A general container for multiple GEMS devices. Creates a “transaction”.	MessageSequenceResponse
GetConfigListMessage	Get a list of the available configurations	GetConfigListResponse
DisconnectMessage	Used to disconnect from a GEMS Device	N/A
	Response for completely unrecognized messages	UnknownResponse

GEMS Base Message Class

- All GEMS Messages Contain
  - GEMS Target
    - Identifies the target device
    - Specified similar to a UNIX path
    - Example: /System1/Device1
  - Message Type
  - Token:
    - Created by the GEMS device during connection
  - GEMS Version
  - Transaction ID
    - Incrementing message number. Request & Reply match
  - Timestamp
    - Timestamp of when the message was created



Message
(GEMS)

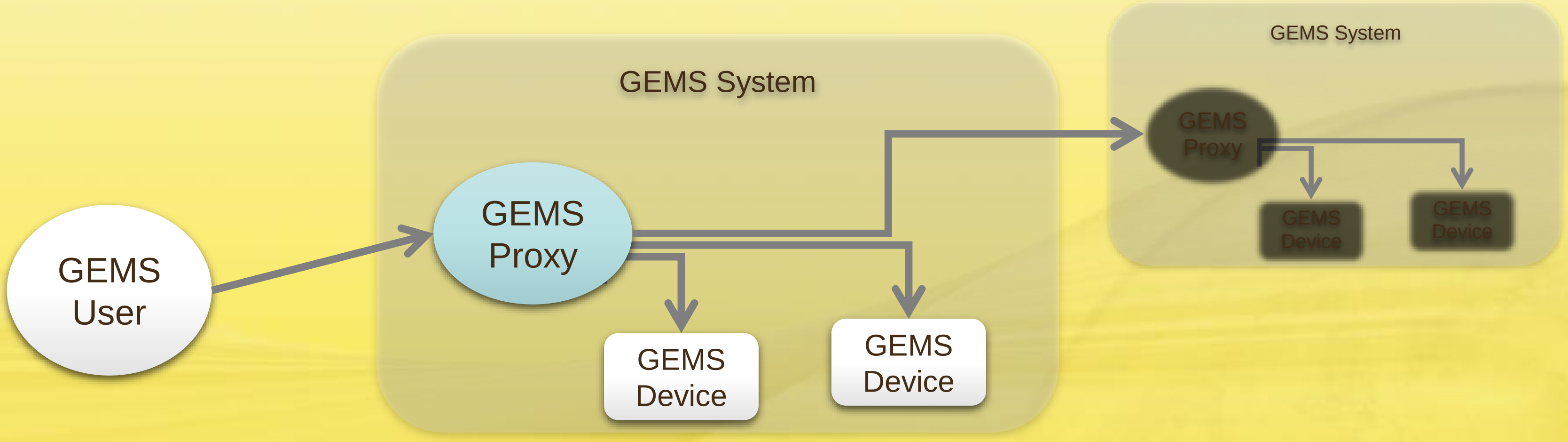
+target : String
+message_type : String
+token : String
+gems_version : int
+transaction_id : int
+timestamp : String

GEMS Target

- Purpose
  - Identify the target GEMS device for the message
  - “Echoed” back in the associated response
- Formatting
  - This is technically a free-text field
  - GEMS recommends using a UNIX style path
    - Example: System1/Device3
    - This provides for hierarchical grouping
    - Future versions may require this to permit connection and message handling at various levels

GEMS Proxy

- The GEMS Proxy Provides For Message Routing
  - Routing is based on the target name
  - Can be internal to a process containing multiple GEMS devices
  - Can be external to the proxy, combining multiple systems
  - Sending a GetConfigMessage to the proxy returns a list of targets
- For TCP and Similar Network Transports
  - The proxy often provides a single socket port for all GEMS Devices



GEMS Token

- Purpose
  - Provide an identifier for each client that could be used to carry additional information about that client
  - Use is optional
  - Provided and encoded by the GEMS Device (Server) as part of a successful connect response
- Formatting
  - This is a free-text field
  - GEMS 1.3 utilizes the token field on the original connect message to provide credentials

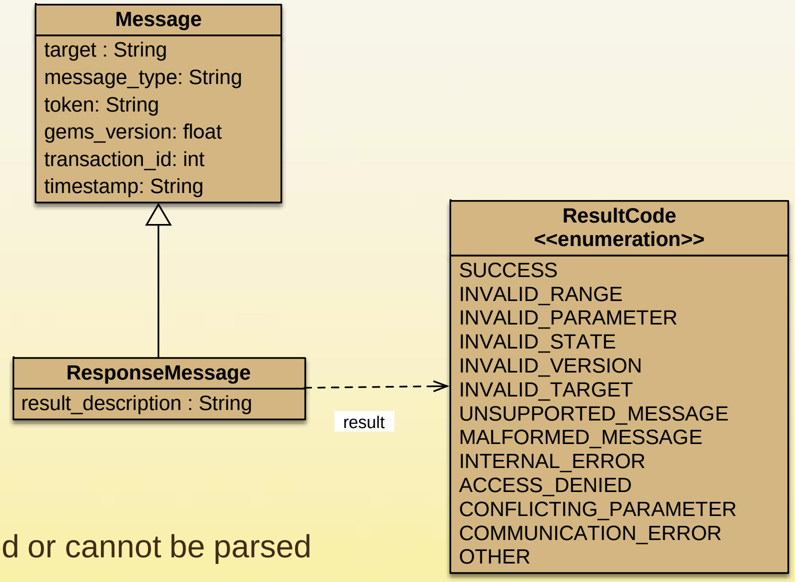
GEMS Versions

- GEMS Versions Are Included In Messages
  - Version 1.2 changed how the version was represented in GEMS-ASCII to better match the actual version.
  - Future plan is to somehow separate the message version from the document version
    - Avoids changing versions when editorial changes are made to the document

GEMS Version	GEMS-ASCII	GEMS-XML
1.0	01	1.0
1.1	02	1.1
1.2	12	1.2
1.3	13	1.3
1.4	14	1.4

GEMS Base Response Message

- Adds The Following
  - Result Description
  - Result Code



Message
target : String
message_type: String
token: String
gems_version: float
transaction_id: int
timestamp: String

ResponseMessage
result_description : String

result

ResultCode
<<enumeration>>
SUCCESS
INVALID_RANGE
INVALID_PARAMETER
INVALID_STATE
INVALID_VERSION
INVALID_TARGET
UNSUPPORTED_MESSAGE
MALFORMED_MESSAGE
INTERNAL_ERROR
ACCESS_DENIED
CONFLICTING_PARAMETER
COMMUNICATION_ERROR
OTHER

- Error Responses
  - Bad Formatting
    - UnknownResponse
      - Used when the original message is not recognized or cannot be parsed
    - MALFORMED_MESSAGE
      - Used when the original message type could be found, but something else was in error (parameter formatting etc.)
  - Other Errors
    - Indicated by the ResultCode and the description

GEMS Result Codes

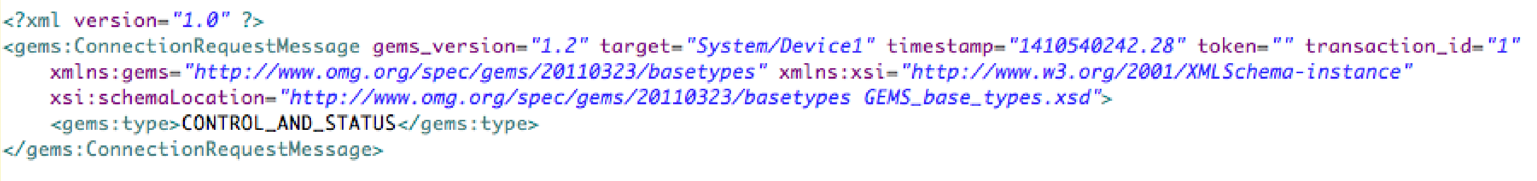
Code	Description
INVALID_RANGE	Indicates that a parameter or argument is out of the acceptable range
INVALID_PARAMETER	Indicates that and unsupported or unknown parameter was provided in the message
INVALID_STATE	Indicates that an invalid state was reached within the GEMS device. This could happen before a connection is established. Might also indicate the state of the device itself. The description should indicate the exact problem.
INVALID_VERSION	Indicates that the GEMS version was unrecognized or unsupported
UNSUPPORTED_MESSAGE	Indicates that the message type is not supported by this device
MALFORMED_MESSAGE	Indicates that some part of the message, typically after the header, was malformed
INTERNAL_ERROR	Indicates that an internal error occurred within the device
ACCESS_DENIED	Indicates that the GEMS user does not have appropriate access to invoke the action defined in the original message
CONFLICTING_PARAMETERS	Indicates that the original message defined a set of parameters that conflict with each other. For example, Mode A might now permit another parameter to be in a specified range.
COMMUNICATION_ERROR	Indicates that some internal communication error occurred. Might be between the proxy and the device or between the GEMS device and an external (hardware) device.
OTHER	Indicates that an error occurred not already defined in the GEMS ResultCodes.

GEMS-XML

- XML Schema Defined Messages
  - Using XML provides “free” parsing and validation
  - Bandwidth is increased due to verbose message structure

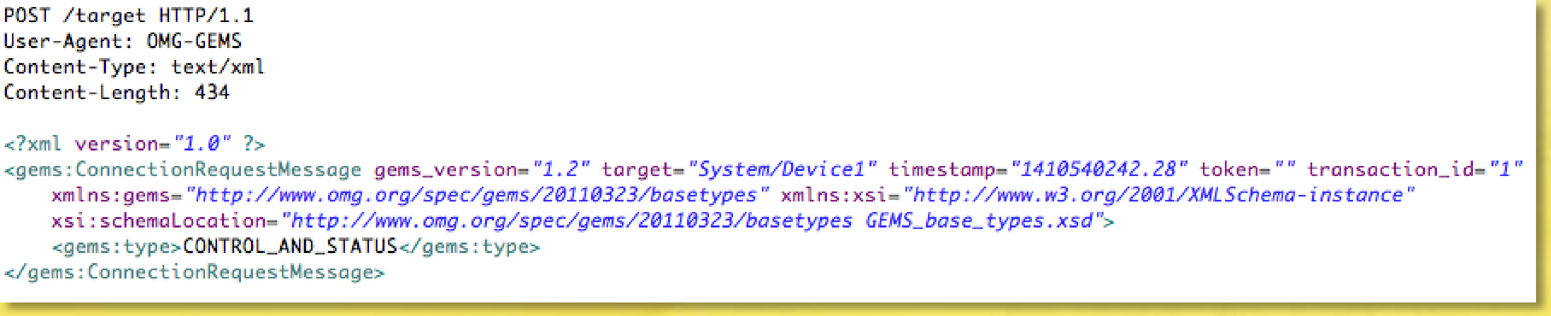
- Schema: GEMS_base_types.xsd

Example GEMS-XML Connect Message



```
<?xml version="1.0" ?>
<gems:ConnectionRequestMessage gems_version="1.2" target="System/Device1" timestamp="1410540242.28" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd">
    <gems:type>CONTROL_AND_STATUS</gems:type>
</gems:ConnectionRequestMessage>
```

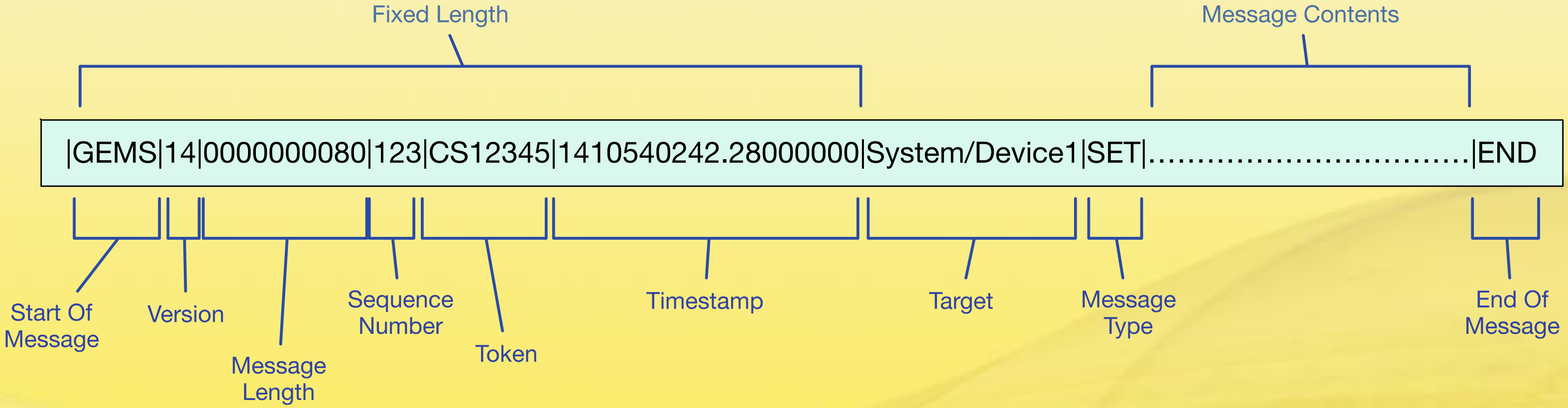
Example GEMS-XML Connect Message With HTTP Header



```
POST /target HTTP/1.1
User-Agent: OMG-GEMS
Content-Type: text/xml
Content-Length: 434

<?xml version="1.0" ?>
<gems:ConnectionRequestMessage gems_version="1.2" target="System/Device1" timestamp="1410540242.28" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd">
    <gems:type>CONTROL_AND_STATUS</gems:type>
</gems:ConnectionRequestMessage>
```

GEMS-ASCII

- Provides A Basic Easy-To-Understand Message Format
  - – Relatively low bandwidth – It is still ASCII though
  - – Manual parsing required
    - Fields are separated by a pipe symbol (vertical bar)



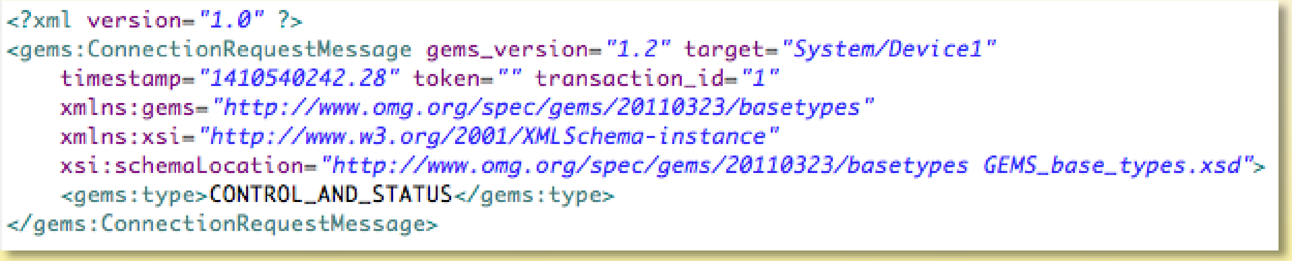
```
|GEMS|14|0000000080|123|CS12345|1410540242.28000000|System/Device1|SET|...............................|END
```

Version 1.4

Connect Message

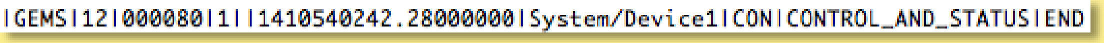
- Connects To A GEMS Device
  - Token field is left empty unless using Authentication

GEMS-XML Example (Version 1.2)



```
<?xml version="1.0" ?>
<gems:ConnectionRequestMessage gems_version="1.2" target="System/Device1"
    timestamp="1410540242.28" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd">
    <gems:type>CONTROL_AND_STATUS</gems:type>
</gems:ConnectionRequestMessage>
```

GEMS-ASCII Example (Version 1.2)



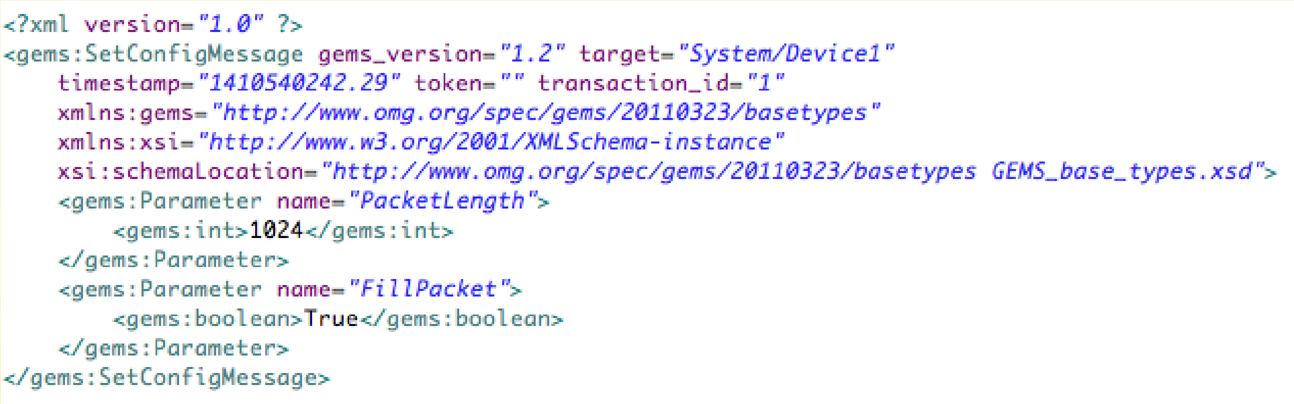
```
|GEMS|12|000080|1||1410540242.28000000|System/Device1|CON|CONTROL_AND_STATUS|END
```

10 characters in the length field for version 1.3+

Set Config Message

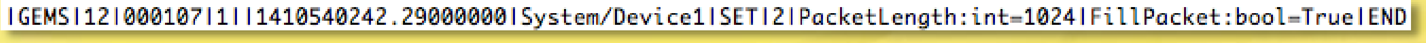
- Sets One or More Parameters
  - Does not need to include all of the parameters

GEMS-XML Example (Version 1.2)



```
<?xml version="1.0" ?>
<gems:SetConfigMessage gems_version="1.2" target="System/Device1"
    timestamp="1410540242.29" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd">
    <gems:Parameter name="PacketLength">
        <gems:int>1024</gems:int>
    </gems:Parameter>
    <gems:Parameter name="FillPacket">
        <gems:boolean>True</gems:boolean>
    </gems:Parameter>
</gems:SetConfigMessage>
```

GEMS-ASCII Example (Version 1.2)



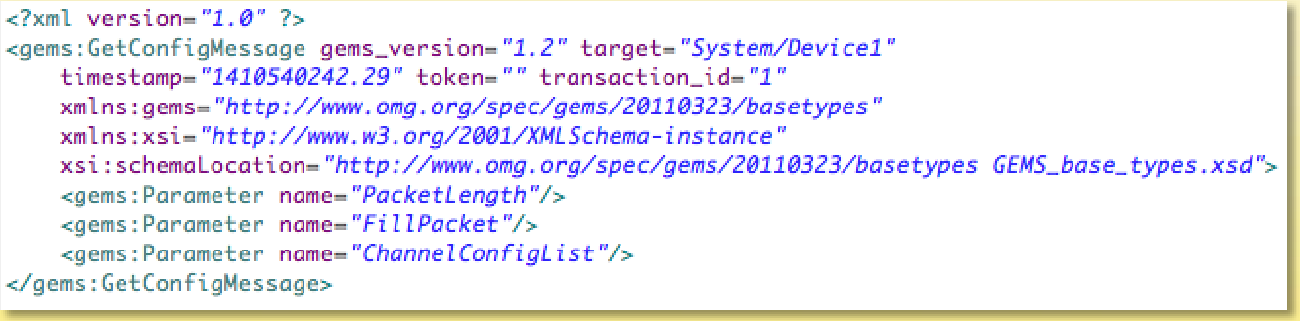
```
|GEMS|12|000107|1||1410540242.29000000|System/Device1|SET|2|PacketLength:int=1024|FillPacket:bool=True|END
```

10 characters in the length field for version 1.3+

Get Config Message

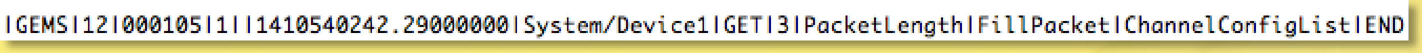
- Retrieve The Current Configuration of the Device
  - Can request specific parameters **
  - Empty list of parameters retrieves all parameters

GEMS-XML Example (Version 1.2)



```
<?xml version="1.0" ?>
<gems:GetConfigMessage gems_version="1.2" target="System/Device1"
    timestamp="1410540242.29" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd">
    <gems:Parameter name="PacketLength"/>
    <gems:Parameter name="FillPacket"/>
    <gems:Parameter name="ChannelConfigList"/>
</gems:GetConfigMessage>
```

GEMS-ASCII Example (Version 1.2)



```
|GEMS|12|000105|1||1410540242.29000000|System/Device1|GET|3|PacketLength|FillPacket|ChannelConfigList|END
```

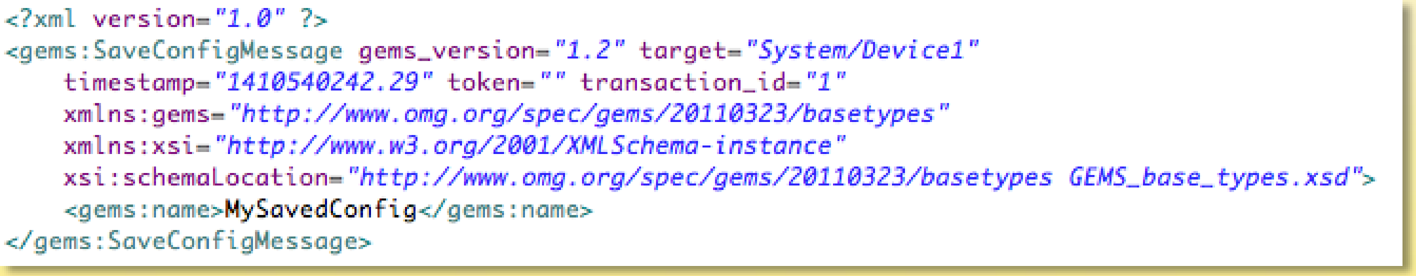
10 characters in the length field for version 1.3+

** Recommend retrieving a specific set of parameters to avoid future additions surprising you

Save Config Message

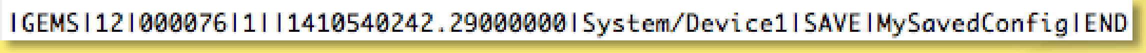
- Saves The Current Configuration of the Device
  - Provide the name of the configuration to save
  - Assumes files can be overwritten
  - NOTE: Often targets the proxy to save full system state

GEMS-XML Example (Version 1.2)



```xml
<?xml version="1.0" ?>
<gems:SaveConfigMessage gems_version="1.2" target="System/Device1"
    timestamp="1410540242.29" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd">
    <gems:name>MySavedConfig</gems:name>
</gems:SaveConfigMessage>
```

GEMS-ASCII Example (Version 1.2)



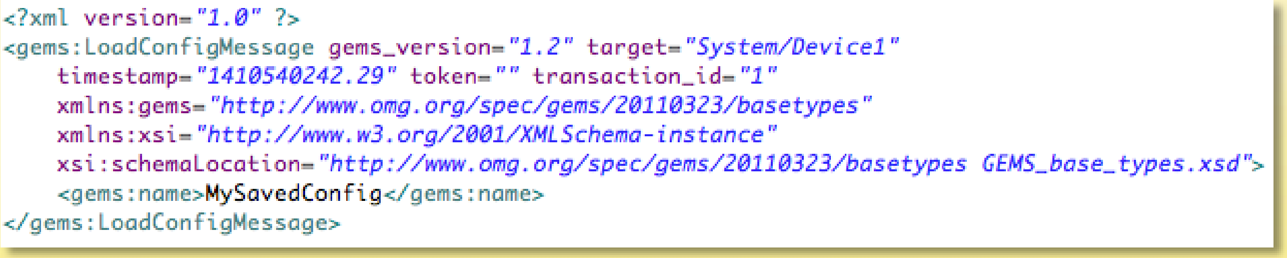
```
|GEMS|12|000076|1||1410540242.29000000|System/Device1|SAVE|MySavedConfig|END
```

10 characters in the length field for version 1.3+

Load Config Message

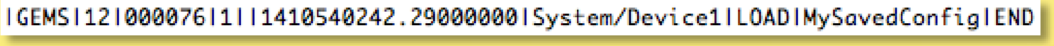
- Loads the Named Configuration File
  - Provide the name of the configuration file to load
  - NOTE: Often targets the proxy to load the full system state

GEMS-XML Example (Version 1.2)



```
<?xml version="1.0" ?>
<gems:LoadConfigMessage gems_version="1.2" target="System/Device1"
    timestamp="1410540242.29" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd">
    <gems:name>MySavedConfig</gems:name>
</gems:LoadConfigMessage>
```

GEMS-ASCII Example (Version 1.2)



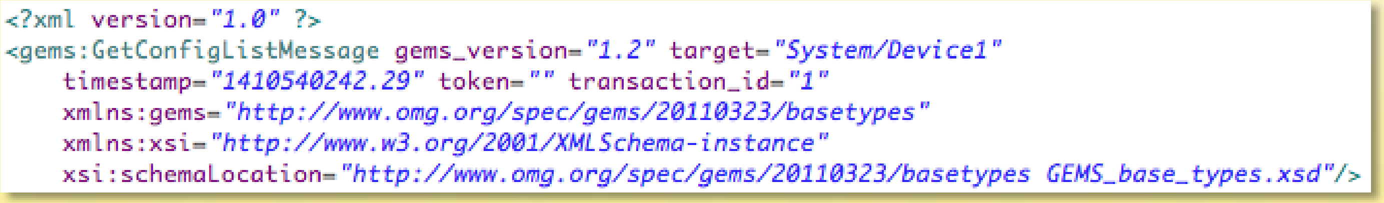
```
|GEMS|12|000076|1||1410540242.29000000|System/Device1|LOAD|MySavedConfig|END
```

10 characters in the length field for version 1.3+

Get Config List Message

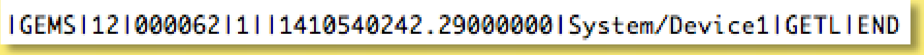
- Retrieve the List of Available Configurations
  - Returns a list of strings

GEMS-XML Example (Version 1.2)



```
<?xml version="1.0" ?>
<gems:GetConfigListMessage gems_version="1.2" target="System/Device1"
    timestamp="1410540242.29" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd"/>
```

GEMS-ASCII Example (Version 1.2)



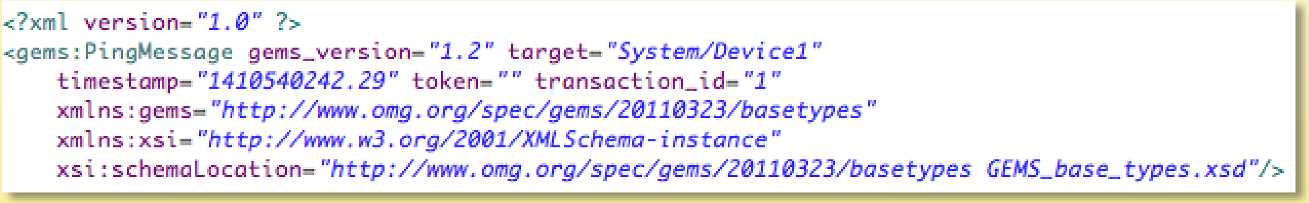
```
|GEMS|12|000062|1||1410540242.29000000|System/Device1|GETL|END
```

10 characters in the length field for version 1.3+

Ping Message

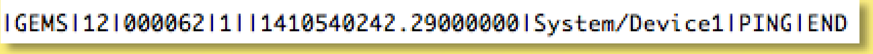
- Simple Message Use to Determine If The Device Is Still There
  - No arguments
  - Helps in a polling environment
  - No change in configuration

GEMS-XML Example (Version 1.2)



```
<?xml version="1.0" ?>
<gems:PingMessage gems_version="1.2" target="System/Device1"
    timestamp="1410540242.29" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd"/>
```

GEMS-ASCII Example (Version 1.2)



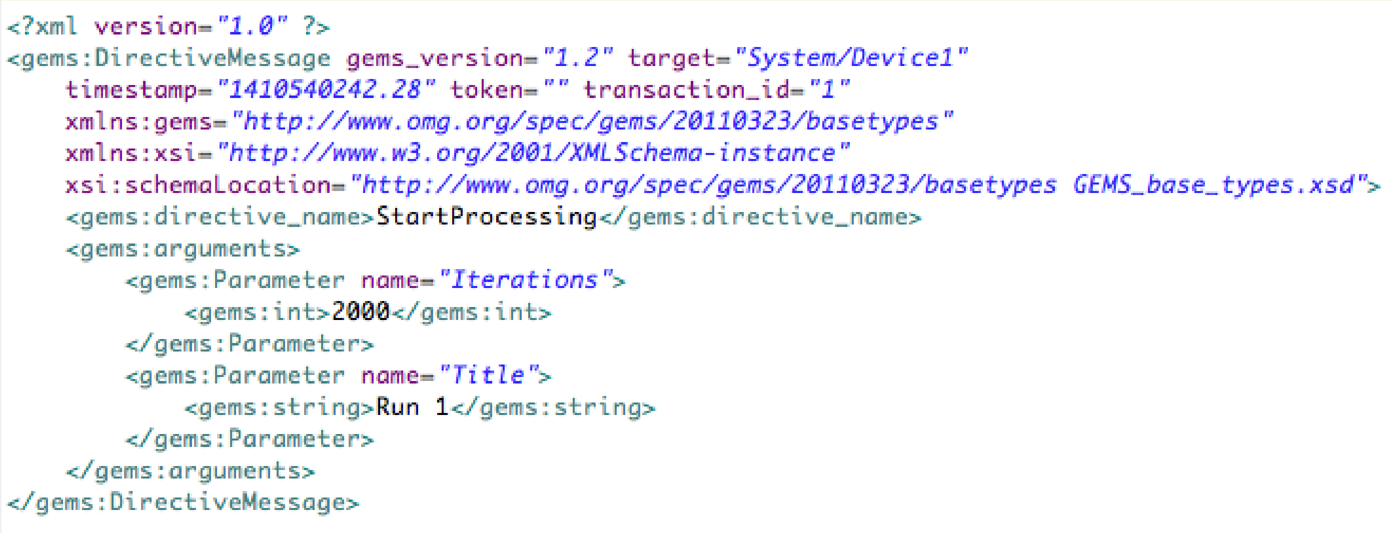
```
|GEMS|12|000062|1||1410540242.29000000|System/Device1|PING|END
```

10 characters in the length field for version 1.3+

Directive Message

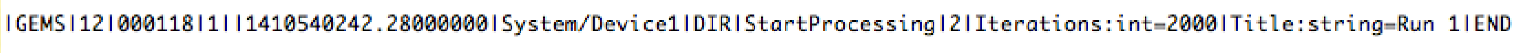
- Used To Invoke an Action
  - Can take any number of arguments: same types as parameters
  - Can return any number of values: same types as parameters

GEMS-XML Example (Version 1.2)



```xml
<?xml version="1.0" ?>
<gems:DirectiveMessage gems_version="1.2" target="System/Device1"
    timestamp="1410540242.28" token="" transaction_id="1"
    xmlns:gems="http://www.omg.org/spec/gems/20110323/basetypes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/spec/gems/20110323/basetypes GEMS_base_types.xsd">
    <gems:directive_name>StartProcessing</gems:directive_name>
    <gems:arguments>
        <gems:Parameter name="Iterations">
            <gems:int>2000</gems:int>
        </gems:Parameter>
        <gems:Parameter name="Title">
            <gems:string>Run 1</gems:string>
        </gems:Parameter>
    </gems:arguments>
</gems:DirectiveMessage>
```

GEMS-ASCII Example (Version 1.2)

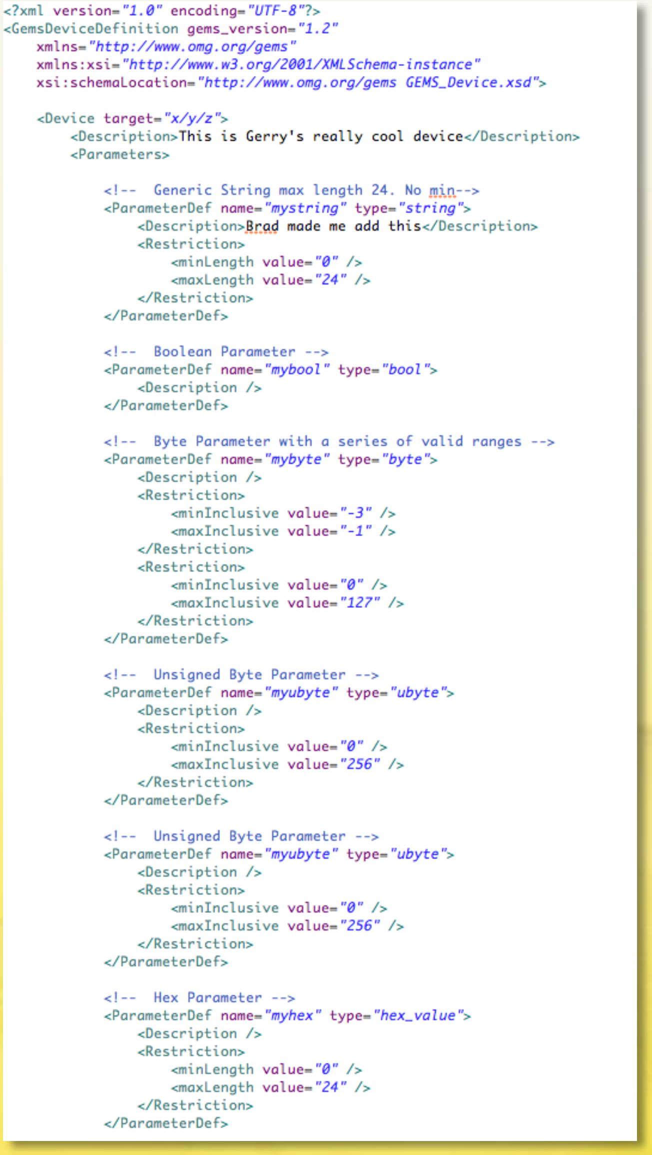


```
|GEMS|12|000118|1||1410540242.28000000|System/Device1|DIR|StartProcessing|2|Iterations:int=2000|Title:string=Run 1|END
```

10 characters in the length field for version 1.3+

GEMS Device Definition File

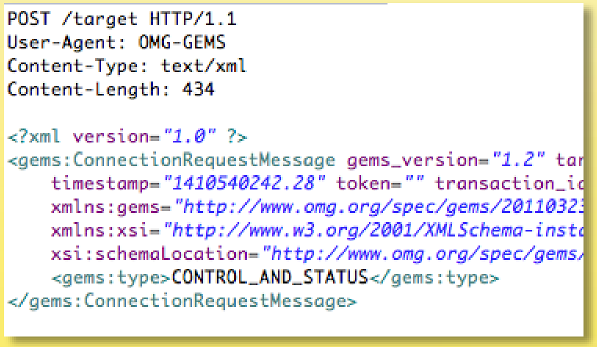
- Purpose
  - The GEMS Device Definition File provide a mechanism for device to describe their capabilities, parameters, ranges, and even add descriptive text
- Formatting is XML
  - This applies to an GEMS Device
  - Not specific to GEMS-XML or GEMS-ASCI
- How a Device Provides This Is Not Yet Standardized
  - Envision web server or other similar capability



```xml
<?xml version="1.0" encoding="UTF-8"?>
<GemsDeviceDefinition gems_version="1.2"
    xmlns="http://www.omg.org/gems"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.omg.org/gems GEMS_Device.xsd">

    <Device target="x/y/z">
        <Description>This is Gerry's really cool device</Description>
        <Parameters>

            <!--  Generic String max length 24. No min-->
            <ParameterDef name="mystring" type="string">
                <Description>Brad made me add this</Description>
                <Restriction>
                    <minLength value="0" />
                    <maxLength value="24" />
                </Restriction>
            </ParameterDef>

            <!--  Boolean Parameter -->
            <ParameterDef name="mybool" type="bool">
                <Description />
            </ParameterDef>

            <!--  Byte Parameter with a series of valid ranges -->
            <ParameterDef name="mybyte" type="byte">
                <Description />
                <Restriction>
                    <minInclusive value="-3" />
                    <maxInclusive value="-1" />
                </Restriction>
                <Restriction>
                    <minInclusive value="0" />
                    <maxInclusive value="127" />
                </Restriction>
            </ParameterDef>

            <!--  Unsigned Byte Parameter -->
            <ParameterDef name="myubyte" type="ubyte">
                <Description />
                <Restriction>
                    <minInclusive value="0" />
                    <maxInclusive value="256" />
                </Restriction>
            </ParameterDef>

            <!--  Unsigned Byte Parameter -->
            <ParameterDef name="myubyte" type="ubyte">
                <Description />
                <Restriction>
                    <minInclusive value="0" />
                    <maxInclusive value="256" />
                </Restriction>
            </ParameterDef>

            <!--  Hex Parameter -->
            <ParameterDef name="myhex" type="hex_value">
                <Description />
                <Restriction>
                    <minLength value="0" />
                    <maxLength value="24" />
                </Restriction>
            </ParameterDef>
```

GEMS Transport Mechanisms

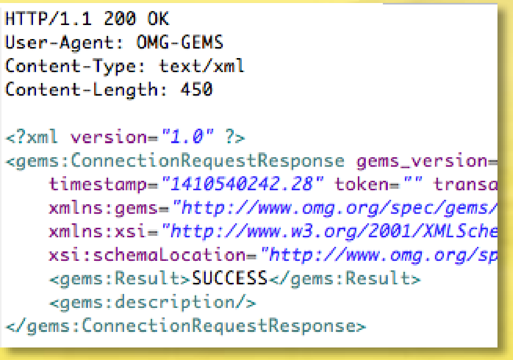
- GEMS Intentionally Does Not Define The Transport Mechanism
  - You can use GEMS across a Network, Secure Sockets, Serial Bus, and many other transports
  - However!
    - We realized the most common transport will be simple TCP streaming
    - GEMS-ASCII is easy – Just send a message, it already includes the packet “framing”
    - GEMS-XML
      - Needed to add a message length field to simplify parsing
      - Decided to use a standard HTTP header. This follows the normal HTTP protocol and is intended to be used by a HTTP library



```
POST /target HTTP/1.1
User-Agent: OMG-GEMS
Content-Type: text/xml
Content-Length: 434

<?xml version="1.0" ?>
<gems:ConnectionRequestMessage gems_version="1.2" tar
    timestamp="1410540242.28" token="" transaction_i
    xmlns:gems="http://www.omg.org/spec/gems/2011032
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
    xsi:schemaLocation="http://www.omg.org/spec/gems/
    <gems:type>CONTROL_AND_STATUS</gems:type>
</gems:ConnectionRequestMessage>
```

HTTP Request

HTTP Response



```
HTTP/1.1 200 OK
User-Agent: OMG-GEMS
Content-Type: text/xml
Content-Length: 450

<?xml version="1.0" ?>
<gems:ConnectionRequestResponse gems_version=
    timestamp="1410540242.28" token="" transa
    xmlns:gems="http://www.omg.org/spec/gems/
    xmlns:xsi="http://www.w3.org/2001/XMLSche
    xsi:schemaLocation="http://www.omg.org/sp
    <gems:Result>SUCCESS</gems:Result>
    <gems:description/>
</gems:ConnectionRequestResponse>
```

GEMS Authentication

- Optional Feature Added In GEMS 1.3
  - Provides for user level authentication
  - GEMS Devices may establish roles and other permissions based on these user credentials
  - It is highly recommend that a secure transport be used (e.g. SSL/TLS)
- Encodes the User Credentials In The Connect Message Token
  - Username & Password
    - up:<username>:<password>

Future Plans

- GEMS 1.5
  - Currently no issues are identified, but that doesn’t mean they aren’t there
  - Please file any issue you find!
- Beyond (all conceptual at this point)
  - Logging & Event Notification
  - Standard Devices
  - Standard Parameters
  - Data Exchange Formats
    - Telemetry and Command Data

For More Information

- http://www.omg.org/space/ - GEMS specification, other SDTF specs
- andzik@amergint.com - OMG SDTF co-chair

SOLM Tutorial

Four-letter acronyms that make a difference in ground system interoperability

GSAW MMXVI

Prepared By The OMG Space Domain Task Force
And

Published by The Aerospace Corporation with permission.

SOLM

- Satellite Operations Language Meta-model

What is a Language Metamodel?

CCL

```
IF (MomentumWheelState .EQ. “Off”) THEN
     CEXL MomemtumWheelOn
ENDIF
```

SpacePython

```
if MomentumWheelState.value() == 'Off':
   sat1.send('MomentumWheelOn')
```

Java

```
if(MomentumWheel.getState().equals(
                             “Off”)){
   sat1.send(“MomentumWheelOn”);
}
```

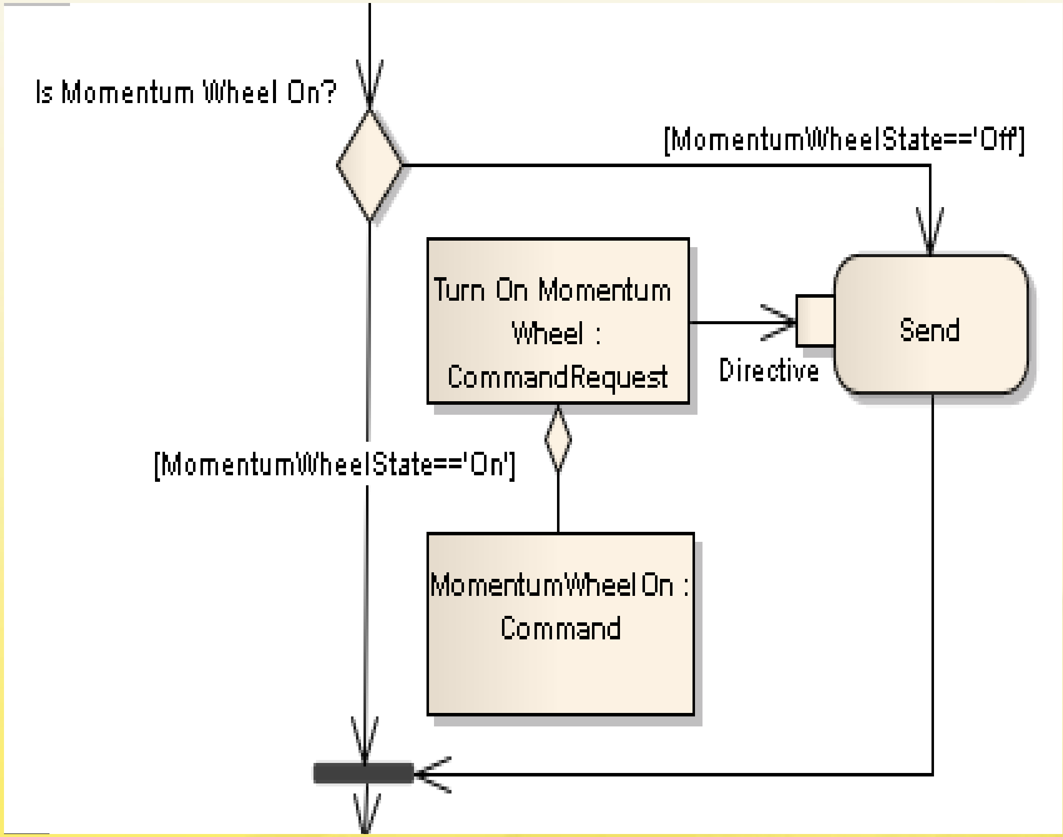
STOL

```
IF (ADMWHLST .EQ. 0)
/CMD MWA, ON
ENDIF
```

Specific languages have a specific syntax with a specific representation of language elements. A language metamodel abstracts the language elements and their relationships so that models can be captured in a form that can be easily translated to different language representations.

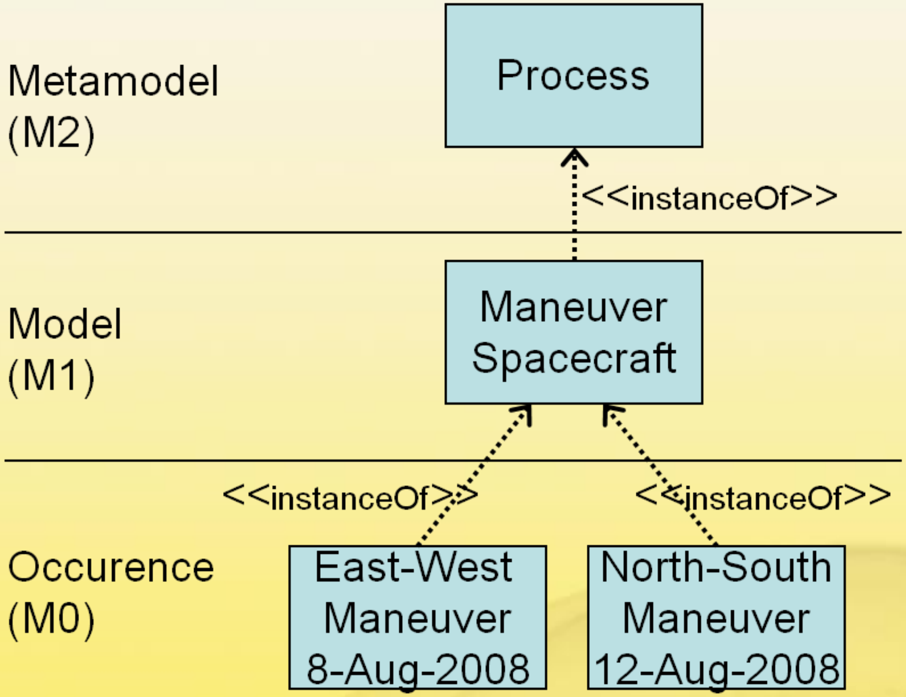
Language is not limited to text

Graphical languages, like the Unified Modeling Language (UML) can also convey procedural information. Here is a small snippet of an activity diagram for checking the state of the momentum wheel and turning it on, if it is currently off.


Is Momentum Wheel On?
[MomentumWheelState=='Off']
Turn On Momentum Wheel : CommandRequest
Directive
Send
[MomentumWheelState=='On']
MomentumWheelOn : Command

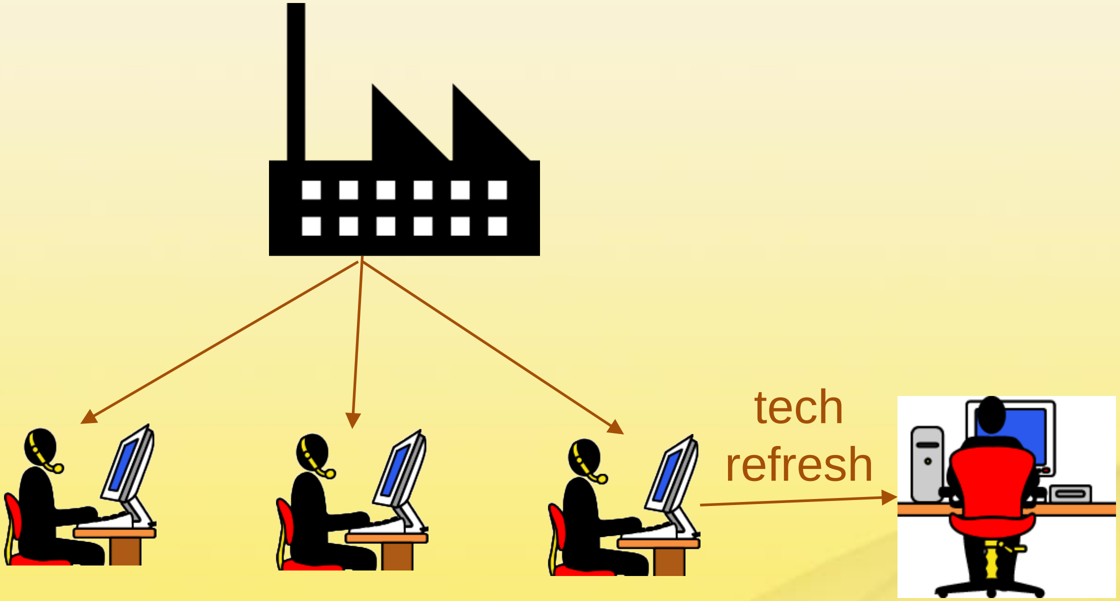
Models, Metamodels, Occurrences

- The SOL Metamodel can capture many abstract procedures (define a process) for many satellites and many ground systems.
- A SOLM Model is a specific operations procedure for a specific satellite, the example here is a procedure to setup a spacecraft maneuver
- An Occurrence is a specific execution of a model (procedure). The record/log of an occurrence is important operations history information



What Are the Benefits of SOLM?

A single procedure definition can be used for satellite control by different ground systems



This allows reuse by cooperating ground systems or transfer of a procedure from an existing ground system to a new one.

Basic Elements of SOLM

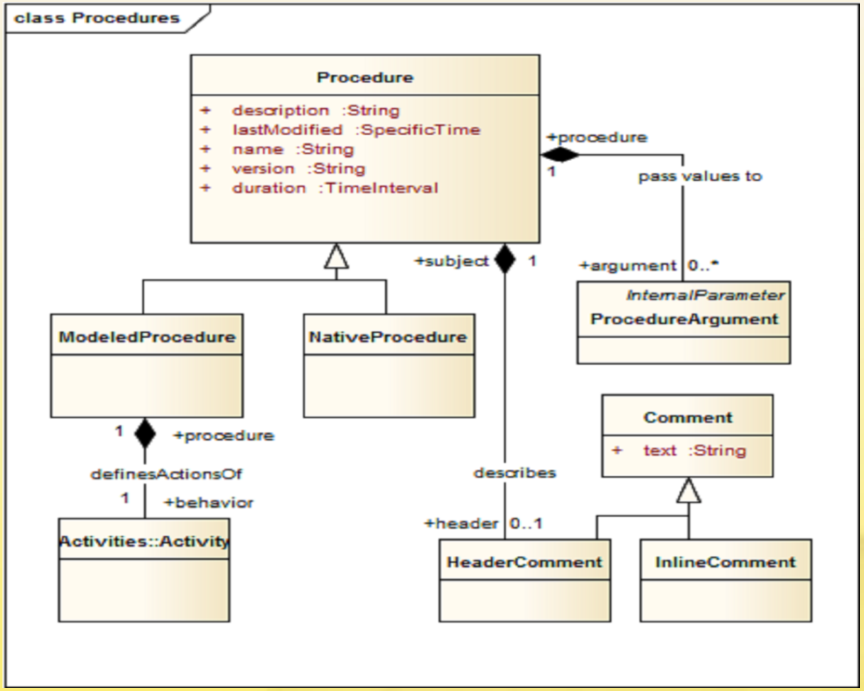
- Procedure
- Parameter
- Procedure Environment
- Activity
- Action

Procedures

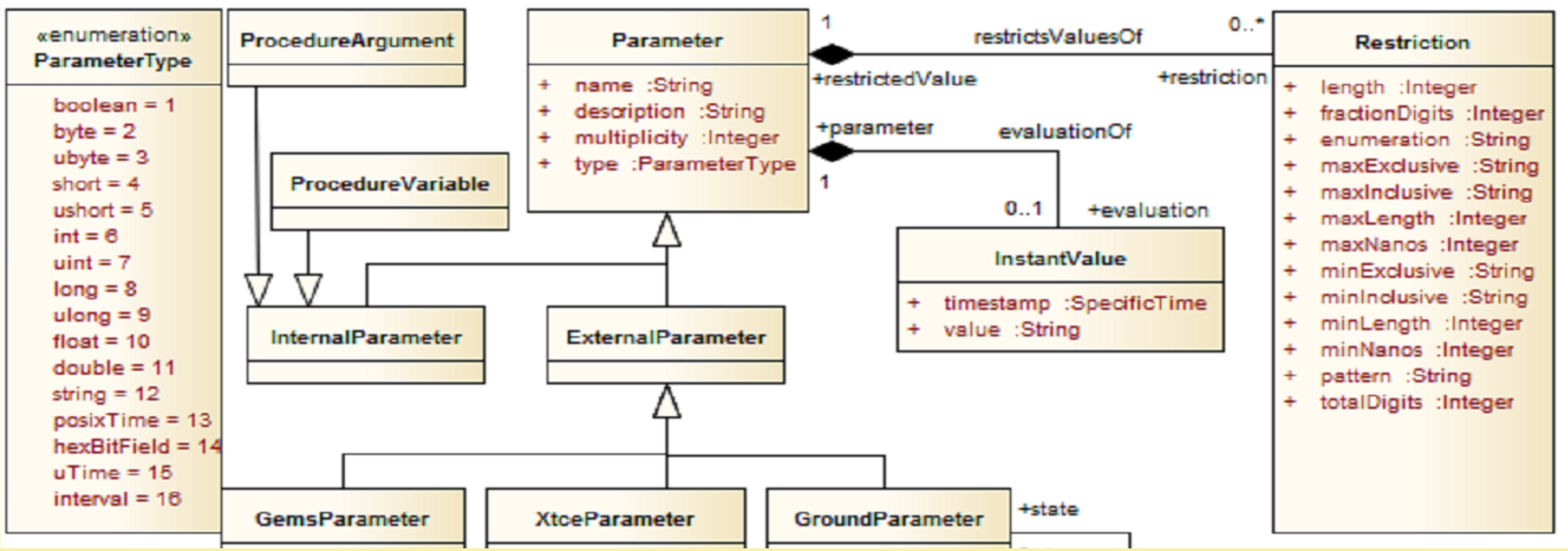
- A procedure is the basic module for satellite operations scripting.
  - Attributes describe the procedure and provide a unique, versioned ID

- Optional arguments supply run-time values to the procedure
  - In-only, return is a single status integer

- “Native” procedures are invoked with parameters by name only, there is no visibility or modeling of the actions



Parameters



- Parameters provide the data used by procedures and allow procedures to produce data outputs

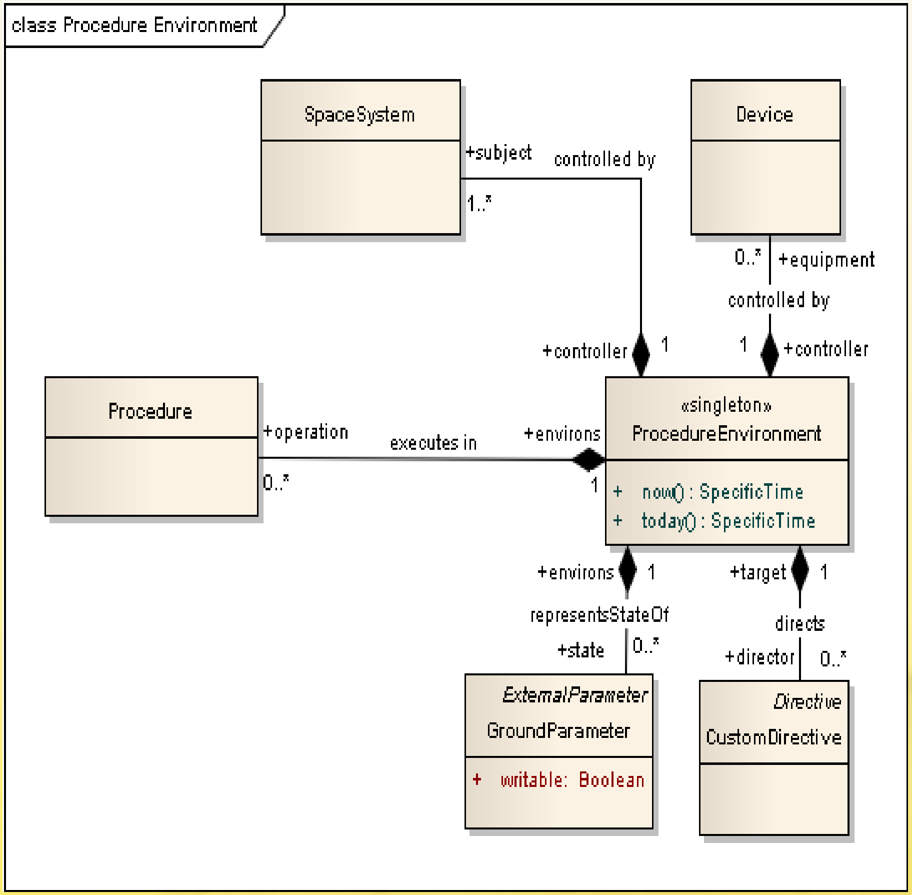
- The set of types was selected to encompass the XTCE and GEMS specifications parameter types.

- The Parameter model is simpler than XTCE. Binary encodings, alarms, and engineering unit conversions are not included.

- Gems and Ground Parameters that are writable may affect the configuration of the device or ground system. Setting the value of an XTCEParameter does not change the state of the SpaceSystem, but can be used as a global Parameter, such as a derived item.

Procedure Environment

- The Procedure Environment represents the Ground System that runs the procedure:
  - The collection of SOLM and native procedures that may be run
  - The collection of ground parameters that may be read and or set by a procedure to determine or change ground system configuration
  - The collection of GEMS devices that are controlled in the ground system.
  - The collection of XTCE SpaceSystems (at least one) that are controlled by the ground system.
  - An optional collection of custom actions defined for the ground system

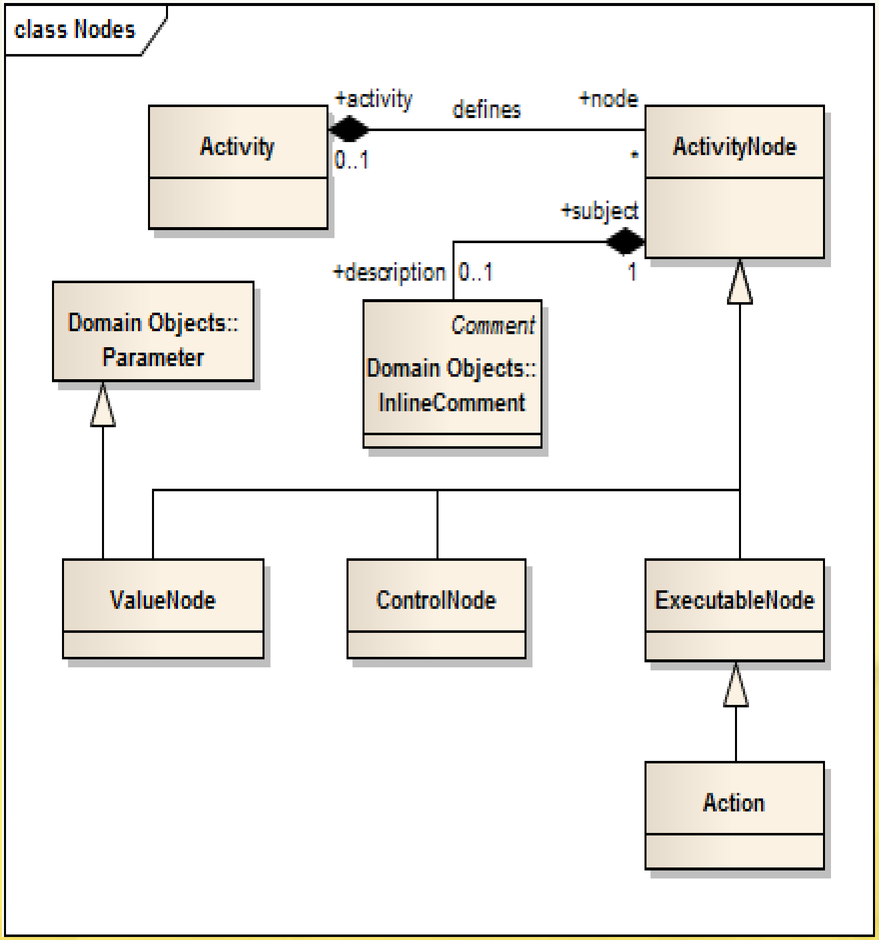


Activity

- An Activity is a collection of nodes and edges that define the control flow, data flow, and actions of a procedure.

- In a model the relationships between the nodes and edges explicitly define data and control flows. A graphical representation (flow chart) visually shows the nodes and edges.

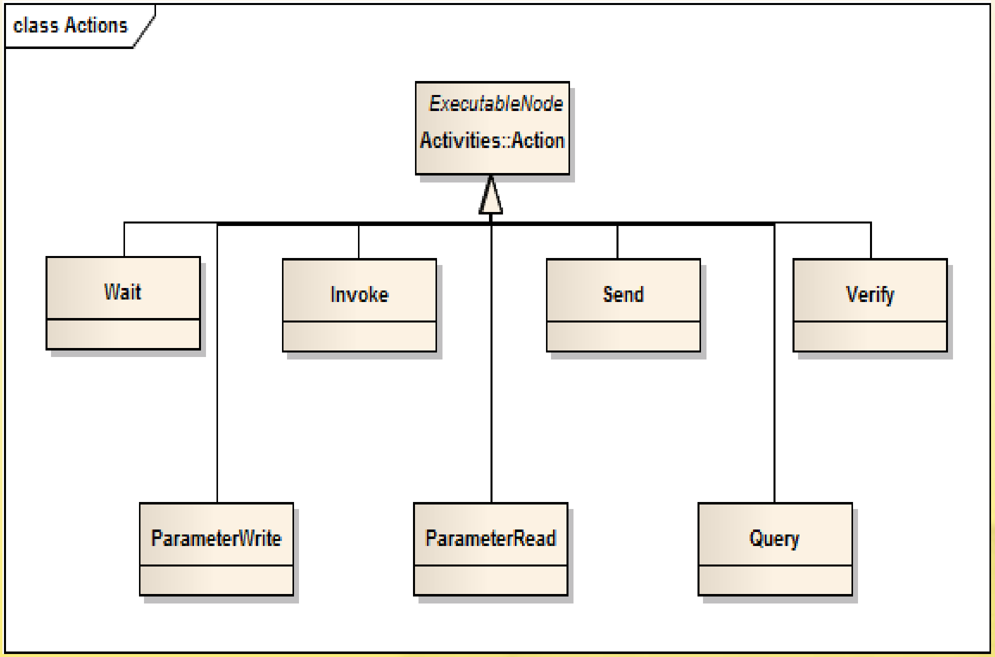
- In a textual language the nodes and edges are defined in the syntax and semantics of the target language.



Action

Actions are activity nodes in a procedure that interact with the ground system and the operator

- Wait – delay procedure execution until an expression is true and/or time elapses.
- Invoke – execute a SOLM or native procedure
- Send – issue a command to a SpaceSystem or GEMS Device
- Verify – that an expression is true or a floating point parameter is equal to a value within a tolerance limit
- ParameterWrite – Set the value of a Parameter
- ParameterRead – Get the current value of a Parameter
- Query – Request a response from the operator



Compliance Levels

- A procedure modeling environment is required to support all SOLM features when defining or importing procedures, in order to be compliant with SOLM.
  - A specific target language (Platform Specific Model in OMG terms) may not support all features, and the modeling environment is only required to provide an error message when generating a procedure in that language.

- An execution environment may support up to three levels of compliance:
  1. Looping, conditionals, timed waits, native procedure invocation, Commands and Parameters from a XTCE document, exception handling for run-time errors.
  2. Support for GEMS device Parameter and Directive definitions.
  3. Multiple threads of execution in a procedure model.

SpacePython?

- SpacePython is an extension package for Python 2.6 that fully maps to SOLM
  - Satellite ground systems is a niche market
  - Python is easily extensible, open source, and has been used in at least two satellite ground systems as the primary scripting language.
    - Perl, Tcl, and Ruby were also candidates
  - Version Control systems work best with text-based programming languages
  - Lower the barrier for SOLM-compliant implementations

Example SpacePython script, 1 of 4

```
#!/usr/bin/python
'''
 This SpacePython module provides an example of a spacecraft
 operations procedure that checks a subsystem state, optionally
 issues a corrective command, and then sends a command to set
 momentum wheel speed based on a change to the current speed.
'''
from space import TimeInterval, SpecificTime, Link
from space import SUCCESSFUL
__version__   = '1.0.0'
__author__    = 'Brad Kizzort'
__copyright__ = 'Copyright 2012, Harris Corporation'
__scriptname__ = 'SetMomentumWheelSpeed'
__duration__   = TimeInterval(0,5) # 5 seconds
__modified__   = SpecificTime(2012, 2, 3, 12, 44)
```

Lets the operating system & editors know this is a python script

Corresponds to HeaderComment in SOLM. Accessible as <module>.__doc__ when module is imported.

__scriptname__, __duration__, and __modified__ are all required. __duration__ may be negative to indicate unknown or variable.

__version__ is required global corresponding to Procedure.version. Accessible as <module>.__version__

Example SpacePython script, 2 of 4

Required function so that all procedures can be invoked with <module>.invoke()

argument is a Namespace object with procedure arguments defined as attributes

Required text corresponds to Procedure.description attribute in the metamodel

```python
def invoke(args):
    '''Change the current momentum wheel speed by the positive
       or negative rpm specified by the keyword parameter,
       SpeedIncrement.
    '''
    sat1 = Link('SAT1')
    MomentumWheelState = sat1.lookupParameter('MomentumWheelState')
    MomentumWheelSpeed = sat1.lookupParameter('MomentumWheelSpeed')
    setWheelSpeed      = sat1.lookupCommand('SetWheelSpeed')
```

A SpacePython procedure must look up commands and telemetry parameters used in the procedure.

Example SpacePython script, 3 of 4

Conditional to check a telemetry value before sending command, note that indentation is significant and required in python.

Send simple command defined in database with no parameters. This references the SpacePython "Link" object which corresponds to a SpaceSystem in the metamodel

```
if MomentumWheelState.value() == 'Off':
    sat1.send('MomentumWheelOn')
setWheelSpeed.setValues(WheelSpeed = \
(MomentumWheelSpeed.value() + args.SpeedIncrement)
sat1.send(setWheelSpeed)
return SUCCESSFUL
```

Sets the value of the "WheelSpeed" CommandArgument in the setWheelSpeed Command that was looked up earlier

Successful procedure return. Can also return a FAILED status.

The send method of a Link can take either a string to do a simple Command lookup, or can take a Command instance with CommandArgument values.

Reference to ProcedureArgument value

Example SpacePython script, 4 of 4

```
from space.parameters import Parameter, MinInclusiveR, MaxInclusiveR
__parameters__ = [Parameter('SpeedIncrement', 'int',
                 restriction=[MinInclusiveR(-10000), MaxInclusiveR(10000)])]
```

This required module global contains a list of the procedure Arguments with attributes describing the datatype and range of values

boilerplate to allow the SpacePython script to be invoked from a shell/cmd window

```
# If invoked from the command line, configure logger, parse arguments, and invoke
if __name__ == '__main__':
    import logging
    import space
    space.log.setLevel(logging.INFO)
    space.log.addHandler(logging.StreamHandler())
    __args__ = space.parseArgs(__scriptname__,__doc__,__parameters__)
    invoke(__args__)
```

Suggestions for Greater Portability of SOLM-based Procedures

- Isolate any user interface code beyond the operatorQuery action in separate procedure modules
  - This includes displays/reports written in the scripting language or menu-style queries.
  - User interface capability varies widely among existing satellite operations languages

- Isolate ground equipment configuration commands in separate modules
  - Ground system capabilities and commands may vary and by isolating that variability, the spacecraft procedures can easily be ported, even if the ground system requires a new setup script

For More Information

- http://www.omg.org/space/ - SOLM specification, other SDTF specs
- https://www.python.org/ - Downloads for Python 2.6 and 2.7
- https://pypi.python.org/ - 51,780 python packages

  (not including SpacePython)
- bkizzort@harris.com -OS/COMET® Systems Engineer,

  SpacePython evangelist , and

  OMG SDTF co-chair