• **Agenda**
  – OMG and SDTF Background
  – XTCE Background
  – XTCE Tutorial
  – XTCE Resources
XTCE
XML Telemetry & Command Exchange

An International (OMG) Standard Language for the Description of Mission Operations Databases
Credits

• Key Design Team
  – European Space Agency
    • Science Systems – Sam Cooper, Roger Thompson
    • ESOC – Mario Merri
  – US Air force/Mission Operations
    • USAF SMC Det 12/VO (Lockheed Martin) – Gerry Simon
  – Boeing Communications & Satellites
    • Janice Champion
  – NASA
    • GSFC – Mike Rackley
    • GST – Kevin Rice
    • Raytheon TSC / UMD – Ed Shaya
More Credits

- OMG Space Domain Task Force
  - Joan Dunham (CSC)(OMG Co-chair)
  - Mark Brucks (J3S)
  - Jeff Condron (GSFC/Raytheon)
  - Greg Greer (GSFC/Hammers)
  - Priscilla McKerracher (APL)
  - Peter Shames (JPL)
  - Brad Kizzort (Harris)
Vision

Satellite Factory

Common Formats Facilitate Transition to Operations and Data Exchange

Mission Ground System

OMG Space Domain Task Force
- because space applications MUST interoperate better

Basic Concepts

- **What**: Non-proprietary standard description language of T&C parameters and commands for exchange between manufacturers, vendors, testers and operators.

- **Why**: Common format reduces human error, ambiguities, confusion, and lost time in difficult, sometimes lossy, conversions between systems.
  - Save money per mission
  - Reduce mission readiness schedule
  - Better, Faster, Cheaper

- **How**:  
  - Common usage of XML to tag information.
  - Greater machine understanding and automation.
Background - Object Management Group (OMG)

- Consortium of ~600 international software companies (Lockheed Martin, IBM, Sun, MS, Borland, etc)
- Standards Organization for CORBA and UML among others. Developed XSI, the XML standard for UML modeling.
- Subdivided by Technology/Industries into platform/domain task forces.
- Meets 5 times per year.

Tasks Forces
- Issue Request for Information/Proposal (RFI, RFP) for specifications.
- Receives responses, then creates a candidate specification.
- Ultimately OMG Architecture Board (votes) recommends specification.
Background - Space Domain Task Force (SDTF)

- The Space Domain Task Force (STDF) is an OMG vertical Domain Task Force (DTF) specifically chartered to foster the development of space related standards because:
  
  Space applications **MUST** interoperate better

  - Space professionals committed to greater interoperability, reduction in costs, schedule, and risk for space applications through increased standardization
  - The SDTF has been working cooperatively with the CCSDS to ensure consistent space standards development.
XTCE - XML Telemetric Command Exchange format

XTCE Encapsulates the data required to perform all this processing.
A Little About XML

- XML is simply:
  \[<\text{TAG}>\text{data}\langle/\text{TAG}>, \text{where data is:}\]
  \[<\text{TAG}>\text{more data}\langle/\text{TAG}>, \text{etc...}\]

- XML is designed to *carry data* – W3C

*Example:*

```xml
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```
From XML to XML Schema

- XML Schema is a LANGUAGE builder for building an information model in XML – a meta-model
- Once you define YOUR Schema, you can build a parser to ingest XMLs. If it accepts the XML, the XML in YOUR Schema... (it’s valid)
- If the XML is in your language, you can then get data into or out of it, after parsing
A Schema for the Last XML Example

<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.w3schools.com"
  xmlns="http://www.w3schools.com"
  elementFormDefault="qualified">

  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
What Does XTCE Do? Data ... MetaData

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
<th>Created by</th>
</tr>
</thead>
<tbody>
<tr>
<td>W3C - Schema</td>
<td>Meta Meta Meta Data – An XML Documents (W3C Schema Language) that describes the format of itself</td>
<td>the W3C</td>
</tr>
<tr>
<td>XTCE – Schema</td>
<td>Meta Meta Data – An XML Document (W3C Schema Language) that describes the XTCE format</td>
<td>the SDTF</td>
</tr>
<tr>
<td>XML in XTCE format</td>
<td>Meta Data – An XML Document (XTCE format) that describes the Data</td>
<td>for a S/C program</td>
</tr>
<tr>
<td>Streams, Containers, Parameters &amp; Commands</td>
<td>Data – Actual streams of bits containing packages of parameters or commands</td>
<td>the S/C, Ground System or Simulator</td>
</tr>
</tbody>
</table>
TrivalSat Example

Telemetry

| Minority Frame 0 | ASM=fa | MFCtr=0 | Bat1V | BeaconStatus |
|------------------|------------------|------------------|------------------|
| Minority 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

<table>
<thead>
<tr>
<th>Command Format</th>
<th>OpCode</th>
<th>Argument(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeaconPwr</td>
<td>OpCode=170</td>
<td>On/Off</td>
</tr>
<tr>
<td>BeaconPwrOn</td>
<td>OpCode=170</td>
<td>On/Off=On</td>
</tr>
</tbody>
</table>
Steps to Building the XML

1. Define a TrivialSat SpaceSystem

TelemetryMetaData

2. Define TelemetryParameterTypes

3. Instantiate Parameters using ParameterTypes

4. Build Sequence Containers

CommandMetaData

5. Define ArgumentTypes

6. Define MetaCommands

7. Define CommandContainers
Quick Notes to help understand the XML

• ‘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
Step 1: TrivialSat - SpaceSystem

XML Standard

  <TelemetryMetaData>
    <ParameterTypeSet></ParameterTypeSet>
    <ParameterSet></ParameterSet>
    <ContainerSet></ContainerSet>
  </TelemetryMetaData>
  <CommandMetaData>
    <ArgumentTypeSet></ArgumentTypeSet>
    <ParameterMetaCommandSet></MetaCommandSet>
    <CommandContainerSet></CommandContainerSet>
  </CommandMetaData>
</SpaceSystem>
Step 2: TrivialSat – ParameterTypeSet

```xml
<ParameterTypeSet>
    <FloatParameterType name="Battery1VoltageType">
        <UnitSet/>
        <IntegerDataEncoding/>
        <DefaultCalibrator>
            <PolynomialCalibrator>
                <Term coefficient="0.125" exponent="1"/>
            </PolynomialCalibrator>
        </DefaultCalibrator>
    </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
Step 3: TrivialSat - ParameterSet

```xml
<ParameterSet>
  <Parameter name="BatteryVoltage" parameterTypeRef="BatteryVoltageType"/>
  <Parameter name="BeaconStatus" parameterTypeRef="BeaconStatusType"/>
  <Parameter name="minorFrameCtr" parameterTypeRef="minorFrameCtrType"/>
</ParameterSet>
```

- Declare a Parameter with this name
- Parameter will be of this type
Step 4: 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
Step 5: TrivialSat - ArgumentTypes

```xml
<ArgumentTypeSet>
  <BinaryArgumentType name="opCodeType">
    <UnitSet/>
  </BinaryArgumentType>
  <EnumeratedArgumentType name="onOffType">
    <UnitSet/>
    <EnumerationList>
      <Enumeration value="1" label="on"/>
      <Enumeration value="0" label="off"/>
    </EnumerationList>
  </EnumeratedArgumentType>
</ArgumentTypeSet>
```
Step 6: TrivialSat – MetaCommand: Top level MetaCommand

Abstract TrivialSatCmd – only has an opCode

```xml
<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="TrivialSatCmdContainerType"/>
   </CommandContainer>
</MetaCommand>
TrivialSat – Complete XML

The entire XML document for TrivialSat
Details

• The remainder of this presentation covers details of the XTCE specification
XTCE: Simple to Complex

- XTCE is designed to describe 90% of the data required for 95% of spacecraft missions
- XTCE for simple data formats is simple
- XTCE for esoteric data formats is not simple
- Pages in this presentation that discuss some of the more esoteric concepts have darker backgrounds
What’s With this Systems-of-Systems Thing?

- XTCE’s hierarchical structure looks like a real spacecraft

- Repeating components may be defined once

- Subsystem data may be independently developed and delivered
More on the hierarchy

- Hierarchy may start at any level

Military Sats
Navigation
GPS
Communication
Weather

OMG Space Domain Task Force
- because space applications MUST interoperate better
<xml version="1.0" encoding="UTF-8"?>
  <TelemetryMetaData>
    <ParameterSet>
      <BinaryParameter name="colorImage">
        <UnitSet/>
        <BinaryDataEncoding>
          <SizeInBits>
            <FixedDecimalValue>1000000</FixedDecimalValue>
          </SizeInBits>
        </BinaryDataEncoding>
      </BinaryParameter>
    </ParameterSet>
  </TelemetryMetaData>
</SpaceSystem>
Telemetry Processing Flow

Stream
stream of ones and zeros, there may be stream level encryption, and/or compression, and/or error coding

FrameStream
a stream that may be blocked into blocks of data, by applying a sync strategy

Container
An instance of a container where (possibly) only the abstract container type is known. The processing system must determine the real Container type using the Restriction element in the universe of sub-types

SequenceContainer
A list of Entry’s. Entry’s may be raw parms, stream segments, containers, or container segments. May include repeating contents where the count is given as a ParmRef. Sequences may be listed using both absolute or relative locations.

Raw Parameter
may use a foreign integer or float format, may be in wrong order, uncalibrated. May also simply be binary data

Application Parm
has a data type, a description, units, limits, etc. In a future version, maybe could be an array or aggregate data structure

StreamProcessing

ContainerProcessing

ParameterProcessing
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 and Boolean

• Simple Case

```xml
<IntegerParameterType name="grdCommandCtr_t">
  <UnitSet/>
</IntegerParameterType>
```
ParameterTypes and Parameters

- ParameterTypes are instantiated as Parameters. Parameters can have a value; it is not the value itself.
- Parameters types indicate how the parameter is to be handled on the ground.
- **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 strings
- **RelativeTime** – an elapsed time value (precision is not currently specified)
- **AbsoluteTime** – time referenced to a fixed epoch
ParameterType – Example XML

```xml
<BinaryParameterType name="frameSyncType" shortDescription="Frame Synchronization Field"
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 – Example XML

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

- **dataSource** – telemetered, derived, constant, local
- **readOnly**
- **SystemName**
  
  *Optional. Normally used when the database is built in a flat, non-hierarchical format*
- **ValidityCondition**
  
  *Optional 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**
  
  *See discussion on following page*
TimeAssociation

- Telemetry parameter instances are oftentimes "time-tagged" with a timing signal either provided on the ground or on the space system. This data element allows one to specify which AbsoluteTimeParameters to use to "time-tag" instances of this Parameter with.

- TimeAssociation is a special case of ParameterInstanceRef that also includes a flag ‘interpolateTime’ that indicates whether the actual value should be used or if the value of the ParameterInstanceRef should be adjusted to reflect the probable time elapsed.
Time values are typically built up from one or more separate Parameter instances starting with least significant and working towards most significant.
Calibrators

- Sensors errors on the spacecraft plus the need to ‘compress’ data for transmission result in a need to ‘calibrate’ incoming telemetry data into usable engineering data.

*Simple Linear Calibration (single coefficient)*
Calibrators – Two Types

- **Polynomial**, any number of coefficients
  \[ Y = C_0 + C_1 x + C_2 x^2 + \ldots + C_N x^N \]
  Where \( Y \) is the calibrated value and \( x \) is the uncalibrated value

- **Spline**, segmented curve, arbitrary order
  - 1\(^{st}\) Order is a set of line segments
  - 2\(^{nd}\) Order is a set of quadratic segments
  - 3\(^{rd}\) Order is a set of cubic segments
  - Etc.
  - 1\(^{st}\) Order is the default
Calibrators – Default and Context

- On rare occasions Calibrations need to change as the SV enters different phases (factory test, on-orbit) or enters different operating modes (eclipse, LEO, safe-mode).
- On other occasions, it may be necessary to apply different Calibration curves depending on the uncalibrated value.
- These phases, and operating modes are collectively called ‘Contexts’.
- ParameterTypes have an optional ContextCalibratorList.
- The Calibrator from the first Context in the ContextCalibratorList to test true is applied.
Packaging - Container UML

NameDescriptionType
- AliasSet : AliasSetType [0..1]{sequenceOrder=2}
- LongDescription [0..1]{sequenceOrder=1}
- name : NameType{use}
- shortDescription{use}

ContainerType
{mixed=false, complexContentMixed=false}
- BinaryEncoding : BinaryDataEncodingType [0..*]{sequenceOrder=3}
- DefaultRateInStream : RateInStreamType [0..*]{sequenceOrder=1}
- RateInStreamSet : RateInStreamSet_AnonymousType [0..*]{sequenceOrder=1}
- RestrictionCriteria : RestrictionCriteria_AnonymousType{sequenceOrder=1}
- containerRef : NameReferenceType{use}

BaseContainer_AnonymousType
{sequenceOrder=2}
- containerRef : NameReferenceType{use}
- RestrictionCriteria : RestrictionCriteria_AnonymousType{sequenceOrder=1}

SequenceContainerType
abstract
- idlePattern : FixedIntegerValueType = 0x0
- ContainerRefEntry
- ParameterSegmentRefEntry

EntryListType
{mixed=false, minOccurs=0, maxOccurs=unbounded}
- ParameterRefEntry
- ContainerSegmentRefEntry

ParameterRefEntryType
- parameterRef : NameReferenceType{use}

IndirectParameterRefEntryType
- aliasNameSpace
- ParameterInstance : ParameterInstanceRefType{sequenceOrder=1}

ArrayParameterRefEntryType
- DimensionList : DimensionList_AnonymousType{sequenceOrder=2}
- lastIndexForThisArrayInstance = false
- parameterRef : NameReferenceType{use}

StreamSegmentEntryType
- order
- sizeInBits{use}
- streamRef : NameReferenceType{use}

ContainerSegmentRefEntryType
- containerRef : NameReferenceType{use}
- order
- sizeInBits{use}

ContainerRefEntryType
- containerRef : NameReferenceType{use}
Packaging – 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.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Obj Ref</th>
<th>Entry</th>
<th>Obj Ref</th>
<th>Entry</th>
<th>Obj Ref</th>
<th>Entry</th>
<th>Obj Ref</th>
<th>Entry</th>
<th>Obj Ref</th>
<th>Entry</th>
<th>Obj Ref</th>
<th>Entry</th>
<th>Obj Ref</th>
<th>Entry</th>
<th>Obj Ref</th>
<th>Entry</th>
<th>Obj Ref</th>
<th>Entry</th>
<th>Obj Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td></td>
<td>location</td>
<td></td>
<td>location</td>
<td></td>
<td>location</td>
<td></td>
<td>location</td>
<td></td>
<td>location</td>
<td></td>
<td>location</td>
<td></td>
<td>location</td>
<td></td>
<td>location</td>
<td></td>
<td>location</td>
<td></td>
</tr>
</tbody>
</table>
Packaging – Entries

- 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
  - The end of the container to the end of this entry
Packaging – 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 0\textsuperscript{th} 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:

![Array Entries Diagram]

- Least significant dimension
- Most significant dimension
Packaging – 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.

<table>
<thead>
<tr>
<th>A₀₀</th>
<th>A₀₁</th>
<th>A₀₂</th>
<th>A₀₃</th>
<th>A₀₄</th>
<th>A₀₅</th>
<th>A₀₆</th>
<th>A₀₇</th>
</tr>
</thead>
<tbody>
<tr>
<td>A₁₀</td>
<td>A₁₁</td>
<td>A₁₂</td>
<td>A₁₃</td>
<td>A₁₄</td>
<td>A₁₅</td>
<td>A₁₆</td>
<td>A₁₇</td>
</tr>
<tr>
<td>A₂₀</td>
<td>A₂₁</td>
<td>A₂₂</td>
<td>A₂₃</td>
<td>A₂₄</td>
<td>A₂₅</td>
<td>A₂₆</td>
<td>A₂₇</td>
</tr>
<tr>
<td>A₃₀</td>
<td>A₃₁</td>
<td>A₃₂</td>
<td>A₃₃</td>
<td>A₃₄</td>
<td>A₃₅</td>
<td>A₃₆</td>
<td>A₃₇</td>
</tr>
<tr>
<td>Aᵢ₀</td>
<td>Aᵢ₁</td>
<td>Aᵢ₂</td>
<td>Aᵢ₃</td>
<td>Aᵢ₄</td>
<td>Aᵢ₅</td>
<td>Aᵢ₆</td>
<td>Aᵢ₇</td>
</tr>
</tbody>
</table>

Entry, size of least significant (last) dimension is 7. 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.
- 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.
Packaging – Sequence Container Inheritance Example

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

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

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

- where FmtID=2

- where MFCtr=21
Packaging - Inheritance

• 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.
Packaging - RestrictionCriteria

- Usually a single Parameter that must be equal to some number – a Comparison
- But, the comparison does not have to be ==
- Or, may be a comparison list (logically anded)
- Or, could be a BooleanEquation
- Or, could have a temporal relationship (container B comes after container A)
- Or, could be a reference to an external algorithm
• ParameterRef, StreamRef, ContainerRef, etc.
• Uses Unix ‘like’ naming across the SpaceSystem Tree
  e.g. /SimpleSat/Bus/EPDS/BatteryOne/Voltage
• The use of an unqualified name will search for a item in the current SpaceSystem first, then if none is found, in progressively higher space systems
  e.g. a parameter reference “VCC” within a Container in the /SimpleSat/Bus sub-system will result first in a search of the Bus sub-system for “VCC” then in /SimpleSat.
• For arrays, use the zero based bracket notation
  e.g. Voltage[3][6];
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.
Data Types

• Parameters and Command Arguments are sub-types of Data
• All data types may have an initial value
Data Encoding UML

- **DataEncoding**
  - Describes how a particular piece of data is sent or received from some non-native, off-platform device. (e.g. a spacecraft)
  - has a bitOrder
  - has a ByteOrderList
  - has an ErrorDetectCorrect

- **IntegerDataEncoding**
  - For all known integer types
  - has a sizeInBits
  - has a type: enum( signMag, unsigned, etc.)
  - has a DefaultCalibrator
  - has a ContextCalibratorList

- **FloatDataEncoding**
  - For common Floating point types
  - has a sizeInBits
  - has a type: enum( signMag, unsigned, etc.)
  - has a DefaultCalibrator
  - has a ContextCalibratorList

- **BinaryDataEncoding**
  - Used for any data that is not in a known format
  - has a sizeInBits
  - has a FromBinaryTransformAlgorithm
  - has a ToBinaryTransformAlgorithm

- **StringDataEncoding**
  - For common String types
  - has a sizeInBits
  - has a type: enum( signMag, unsigned, etc.)
  - has a DefaultCalibrator
  - has a ContextCalibratorList

- **TimeDataEncoding**
  - For common Time data types

- **CCSDSBinary**
  - Type specified in PCode
  - has an Epoch: date - default TAI or UsePField

- **CCSDSASCII**
  - For common CCSDS ASCII types
  - type enum( month/dayOfMonth, dayOfYear )
  - has a hasATerminator
  - has a hasYearSubField
  - has a hasMonthSubField
  - has a hasDaySubField
  - has a hasHourSubField
  - has a hasMinuteSubField
  - has a hasSecondSubField
  - has a numberOfDecimalSecondsDigits

- **NASAPB-5J**
  - For common Time point types
  - has a type: enum( 5JA, 5JB, 5JC, 5JD )
  - has a hasAPCode

- **User**
  - For user defined time codes types
  - has a sizeInBits
  - has a translationAlgorithm

- **CCSDS time encoding is defined in: CCSDS 301.0-B-3**
  - TAI Epoch is 1958 January 1

- **CCSDSBinary - CUC**
  - CCSDS Unsegmented time Code
  - For common Time point types
  - has a type: enum( CUC, CCS, CUC, UsePField )
  - has a size: enum( 1byte, 2bytes, 3bytes, 4bytes, UsePField )
  - has an Epoch: date - default TAI or UsePField
  - has a hasAPCode

- **CCSDSBinary - CCS**
  - CCSDS Calendar Segmented time code
  - Segmented BCD: 16 bits year, 8 bits month, 8 bits day of month, 8 bits hour, 8 bits minute, 8 bits second. Optional resolution.
  - has a calendarVariation: enum( monthOrYear/dayOfMonth, dayOfYear, usePField )
  - has a resolutionSegments: enum( none, 1, 2, 3, 4, 5, 6, usePField )
  - has a hasAPCode

- **CCSDSBinary - CDS**
  - CCSDS Day Segmented time code
  - 16 or 24 bit day segment (from epoch), 32 bit ms of day field, optional 16 (microsecond) or 24 (picosecond) bit submillisecond field.
  - has a type: enum( CUC, CDS, CCS, UsePField )
  - has a hasSegmentLength: enum( 2bytes, 3bytes, UsePField )
  - has an Epoch: date - default TAI or UsePField
  - has a hasAPCode
• XTCE does not have its own algorithm languages, 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.
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.
Alarms – Simple to Complex

• Simple (Default Limits):
  – *warning, critical*

```xml
<DefaultAlarm>
  <StaticAlarmRanges>
    <WarningRange minInclusive="-22" maxExclusive="12"/>
    <CriticalRange minExclusive="-40" maxInclusive="13.7"/>
  </StaticAlarmRanges>
</DefaultAlarm>
```
Alarms – A Little More Complex

• Actually 6 ranges are available in increasing 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
Alarms – ChangePerSecond

• Parameters have an optional ChangePerSecond (Dynamic/rate-of-change) set of alarm ranges
• Both have static limits and rate-of-change limits

```xml
<DefaultAlarm>
  <ChangePerSecondAlarmRanges>
    <WarningRange minInclusive="5" maxExclusive="5"/>
    <SevereRange minExclusive="-8" maxInclusive="8"/>
  </ChangePerSecondAlarmRanges>
</DefaultAlarm>
```
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.
Alarms – Context Alarm Example

```xml
<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 – ConditionalAlarm

- 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
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 - Numeric Alarm UML

ContextNumericAlarmConditionList
Quite often a spacecraft will have different alarm limits when in eclipse, or LEO or in other operational stages. These are called Contexts.
- has an ContextAlarmList : ContextNumericAlarm[]

ContextNumericAlarmCondition
Quite often a spacecraft will have different alarm limits when in eclipse, or LEO or in other operational stages. These are called Contexts.
- has a Context : MatchCriteria
- has an AlarmCondition : AlarmCondition

NumericAlarmCondition
An abstract type
- has a minViolations = 1
- has a (StaticAlarmRange and DynamicAlarmRange) or CustomAlarm or ConditionalAlarm

AlarmRanges
Contains five ranges: Watch, Warning, Distress, Critical, and Severe each in increasing severity. Normally, only the Warning and Critical ranges are used and the color yellow is associated with Warning and the color red is associated with Critical. The ranges given are valid for numbers lower than the min and higher than the max values. These ranges should not overlap, but if they do, assume the most severe range is to be applied. All ranges are optional and it is quite allowed for there to be only one end of the range.
- has an appliesToCalibratedValues
- has a WatchRange : DecimalRange
- has a WarningRange : DecimalRange
- has a DistressRange : DecimalRange
- has a CriticalRange : DecimalRange
- has a SevereRange : DecimalRange

CustomAlarm
An external algorithm that gets called everytime the parameter updates. Algorithm must return an integer corresponding to the severity
- has a minViolations : nonNegativeInt
- has an AlgorithmRef

ConditionalAlarm
Usually a boolean equation
- has a WatchRangeMatchCriteria
- has a WarningRangeMatchCriteria
- has a DistressRangeMatchCriteria
- has a CriticalRangeMatchCriteria
- has a SevereRangeMatchCriteria

StaticAlarm
Alarm condition that triggers when the value exceeds or goes under certain values

DynamicAlarm
Alarm condition that triggers when the value changes too fast (or too slow)
- has a timeInSeconds : decimal default = 1.0

Severity terms:
- watch
- foreboding
- concern
- caution
- warning
- trepidation
- distress
- critical
- severe
- destructive

OMG Space Domain Task Force
- because space applications MUST interoperate better

Copyright © 2003 Object Management Group.
Command Processing Flow

• Similar to Telemetry processing flow
Command Execution Sequence

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

Transmission Warnings
Used to signify the implications of the 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

FAIL
Command Verification Stages

The Command link may simply be earth to space, but it may also involve several ‘hops’ before it reaches the target spacecraft.

- The Command has reached the receiving end of this link
- The Command has been sent on this link
- The command has passed any authentication by the target has been excepted for execution
- The command is now queued for execution. It may be queued for execution at a much later time.
- The command is being executed.
- The command is complete.
Command Restrictions

- Prohibited - Command should not be sent
- spaceSystemRisk – A warning that the issuance of this command may incur risk to the space system
- missionRisk – A warning that the issuance of this command may disrupt service
- Caution – A general warning
Nuances - Descriptions

- shortDescription – all major elements have this attribute for short “tool tip” size descriptions (max size of 32 characters)

- LongDescription – all major elements also have this as a sub-element; it is intended for “instructive” type descriptions. HTML markup is allowed in LongDescriptions.  

*Neither shortDescription nor LongDescription have size limitations.*
• Aliases – all major Elements may have an unlimited number of aliases. Aliases may be used for ground IDs, Onboard IDs, mnemonics etc.

• An alias has a nameSpace e.g., *OnBoardID* and the alias e.g. *020660*

```xml
<AliasSet>
  <Alias nameSpace="Mnemonic" alias="BAT1I"/>
  <Alias nameSpace="OnBoardID" alias="020660"/>
</AliasSet>
```
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 – Physical Address

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

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

\[
\text{Joules} = \frac{\text{Joules}}{m^4}
\]
Schema Style Notes Used By The SDTF

- Element and Type names begin with a capitol letter.
- Type names end with the word "Type".
- Attribute names begin with a lowercase letter.
- Usually, when the UML class diagram references classes, W3C Elements are used, and whenever the UML references simple types (strings, ints), W3C Attributes are used. In general, attributes are preferred over elements because they're easier to deal with in SAX and DOM, but whenever the Element/Attribute may one day carry metadata, elements should be used. One exception, is enumerated classes, because enumerations may be defined for attributes but not for elements.
- Bias toward self-describing names over short, bandwidth conserving ones.
- Use mixed case in names rather than underscores to combine multiple words (camelCase).
Schema Style Notes (cont)

- A documentation annotation is included in every element and type definition. Annotations for a type are included with the type definition, use of the type are annotated in the element definition.

- Hints on units (for values with units) are provided in the names of attributes and elements (e.g. "dataRateInBPS" is preferred over "dataRate" OR "frameLengthInBits" is preferred over "frameLength").

- Major elements or any elements used multiple times are first defined with a complexType definition.

- All collections are put inside either a "List" element or a "Set" Element depending on whether the collection is ordered or unordered.

- Simplicity in compliant XML files is favored over simplicity in the schema.
Resources

- SDTF - http://www.omg.org/space
- Xerces - http://xml.apache.org/#xerces
- Castor - http://www.castor.org/
Similar in concept to “Unix” system services. For example TCP port 80 always has HTTP packets under Unix. So on your spacecraft, APID 1024 might always have memory dump telemetry packets. Separating that out as a “service” could help you organize your telemetry. (blame ESA/PUS for this... but seems like a good idea)
• There is some debate on the necessity of this.