ABSTRACT

In order to help manage the complexity in designing and configuring network-based telemetry systems, and to promote interoperability between equipment from multiple vendors, the integrated Network-Enhanced Telemetry (iNET) Metadata Standards Working Group (MDSWG) has developed a standard language for describing and configuring these systems.

This paper will provide the community with an overview of Metadata Description Language (MDL), and describe how MDL can support the description of the requirements, design choices, and the configuration of devices that make up the Telemetry Network System (TmNS). MDL, an eXtensible Markup Language (XML) based language that describes a TmNS from various aspects, is embodied by an XML schema along with additional rules and constraints. Example MDL instance documents will be presented to illustrate how MDL can be used to capture requirements, describe the design, and configure the equipment that makes up a TmNS. Various scenarios for how MDL can be used will be discussed.

KEYWORDS

iNET, Metadata, XML, MDL, schema

INTRODUCTION

The integrated Network-Enhanced Telemetry (iNET) project is developing standards for network-based telemetry systems to promote interoperability between systems, devices, and applications which may have been developed by different organizations and vendors. The project is developing new capabilities for telemetry systems, including a two-way wireless communications link between test articles and ground stations, as well as a system level management capability for configuring, controlling, and monitoring the systems and devices.
With the increases in capability necessarily comes an increase in complexity in these systems. Additional technologies and devices require management. Managing the increased complexity requires the use of applications that can automate the configuration, control, and status monitoring of the equipment and systems. For this type of automation to be feasible requires standard management interfaces and configuration information. The iNET project recognized the need for such standards and created the system management standards working group (SMSWG) and the metadata standards working group (MDSWG).

The SMSWG has developed standard interfaces for managing systems and devices. The system management interface standard is based on simple networking management protocol (SNMP) and supports the management of faults, configuration, accounting, performance, and security (FCAPS). The configuration interface depends upon a metadata file that will be used to configure systems and devices. A standard format of the configuration file is required to ensure interoperability.

The MDSWG has developed metadata description language (MDL), a common interchange language that describes requirements, design choices, and configuration information for Telemetry Network Systems (TmNS). MDL is an eXtensible Markup Language (XML)-based language, the syntax of which is constrained by a schema. The MDL schema itself is described using an XML schema. The MDL schema is specified in the XML schema definition (XSD) language, which is defined by the world wide web consortium (W3C) (www.w3.org). A configuration file is an instance of the MDL language, and is called an MDL instance document.

The MDL language syntax consists of a vocabulary, rules for construction, and a set of types. The syntax provides a guide for building sentences in the MDL language to describe telemetry systems. Sentences in the MDL language are stored in instance documents. The syntax is encoded into the MDL schema, which can be used to check a specific instance document for compliance with the syntax of the MDL language.

Language syntax provides the basic structure of sentences, but does not define the meaning of those sentences. The semantic of MDL is the meaning of the network and data descriptions, which is driven by how those descriptions will be used in the test process. MDL instance documents will be constructed using tools during the design and implementation of tests. Sections of the metadata instance documents will be created using potentially range and/or test specific workflows, and pieces of metadata will perhaps be reused from earlier tests. The rules that govern whether a document is valid must be designed in a way that is cognizant of the current state of the document in the design process. This presents a challenge, as schema based syntax checks do not provide the richness necessary to define sophisticated context-sensitive rules. These rules must be developed based on context-sensitive semantics that follow from the ways the document is likely to be used in the important stages of development.

The semantic of MDL is represented by the set of use cases, and the transformations that will be performed on the instance documents during the test. The semantic is strongly interrelated to document validity and the use cases, and will be encoded into the tools that implement a particular workflow. The biggest challenge in developing a standard language lies not in
defining the syntax, but in creating a common semantic understanding so that individuals, applications, and tools will be able to exchange information non-ambiguously.

This paper will explain MDL in detail. It will begin by explaining the complexities that drive the need for a common metadata language. Following is a brief discussion of representative use cases and how the MDL language semantic will support them. It concludes with a description of the MDL syntax, guided by a set of examples that will show how MDL can be used to support the required scenarios.

**COMPLEXITY IN NETWORK-BASED TELEMETRY SYSTEMS**

Test instrumentation is evolving to meet the growing demand for test data. This demand is driven by more complex systems being tested as well as the validation of higher fidelity models that are part of the design processes used to develop modern air vehicles and other remotely tested platforms.

To accommodate these needs, commercial networking technologies are being adapted and incorporated into traditional acquisition hardware to allow measurements at the edges to be aggregated for onboard recording and transmission to live test monitoring sites. In fact, many of the systems being tested are implemented with networking technologies that potentially support a very large number of measurements.

These networked acquisition systems are more complex than their predecessors and so require more data to describe their configurations, as well as that of the larger test environment.

Operation of these networked acquisition systems requires large amounts of descriptive information to be captured and exchanged by various software applications and involve human interaction with the engineers that use, select, and install them. The various roles performed by participants in the testing activities dictate which subsets of the descriptive data are of interest.

The volume of descriptive data required, coupled with the role-oriented sub-setting, highlights the benefits of creating special tools to facilitate interaction with the data to minimize human error and ease the management of complex relationships within the data itself.

These tools often collaborate without foreknowledge. A standard descriptive language is required to enable collaboration in an extensible manner whereby new interactions and functions can be achieved without degrading existing ones. For this type of future proofing to actually occur, the descriptive language must be designed up-front with this flexibility in mind.

A model-driven approach was used to develop MDL, which allowed a system level view of the metadata. This set the stage for correctly choosing the boundaries between abstraction and simplicity, while keeping the door open for extensibility to unforeseen future capabilities.
MDL USE SCENARIOS

For the purposes of this paper, a notional workflow for the design of an instrumentation system will be described. This workflow will be separated into scenarios for each step. It is not intended for this workflow to be complete, but only to serve as a basis for highlighting several key features of the MDL.

Describe Test Requirements: The design of the instrumentation is initiated as a set of measurement descriptions from engineers and analysts that will evaluate the data that is ultimately collected during the test. These descriptions may include how the measurement is to be acquired (sensor installation or monitor an existing bus), the expected range of the measurements to be taken (in engineering units), the desired frequency content, the uncertainty required, the classification level, how the measurement should be available during the test (e.g. telemetered and recorded), and so on.

Describe Data Formats: Data formats are selected that are capable of satisfying the requirements. Essentially this step is where the “data containers” are described. This is primarily with respect to the sample rates chosen to achieve either adequate frequency content for analog measurements or to ensure that no messages are lost for bus monitors. Selecting these formats is never quite that simple, however, and in reality, requires many trades to be evaluated to ensure that the data can be transported and processed by available hardware. Guiding principles are also considered, such as attempting to minimize the overhead associated with just moving the data through the system.

Design Test Network: Once the acquisition hardware and data formats have been selected, the pieces must be assembled into a system. The result is a description of how everything is interconnected. There are many trades, including capacities of each network link, switch, router, etc. Here, too, there are guiding principles that are sometimes at odds such as the desire to minimize the number of connections balanced against the physical distribution of the sensor locations. There may also be second-tier issues like creating virtual networks to achieve some specialized function.

Optimize the Network: This is the area where all of the pieces come together and where the most new concepts will be encountered by instrumentation engineers familiar only with the current PCM-based systems. Practical limitations of the commercially derived devices that comprise the network must be balanced against performance expectations in terms of how measurements are delivered to the engineers and analysts during the conduct of the test. Instrumentation engineers will choose how to group measurement packages into messages and assign markings for those messages that indicate how they should be handled by devices in the network relative to other messages.
MDL SYNTAX AND EXAMPLES

MDL describes telemetry systems from several aspects. The primary aspects include data descriptions, quality of service descriptions, and network descriptions.

Data Descriptions: MDL includes five major data-related descriptions:

- Measurements
- Packages
- TmNS data messages (TDM)
- Transport attributes
- Paths

XML snippet A in Figure 1 shows an example of a <TransportAttributes> element describing data available on a Network.

Figure 1. MDL Descriptions of Data

The large blue line in Figure 1 represents the Path (as described by the <Path> element in snippet C), which is a collection of references to Network connections. The small green and orange lines transposed on the blue line represent the two TmNSDataMessages sent using the described transport attributes.
Figure 2 depicts the encapsulation of data as described in the MDL. `<TransportAttributes>` elements (describing UDP/IP settings) contain references to contained `MessageDefinition` elements (describing contained `TmNSDataMessages`). One or more `TmNSDataMessages` may be referenced by a set of transport attributes.

The `<MessageDefinition>` element describes a `TmNSDataMessage`. An example of `<MessageDefinition>` XML can be seen in snippet B in Figure 1. This description may include maximum message latency or length, a message definition identifier, a message length, and several other concepts. A `<MessageDefinition>` element contains references to `<TransportAttributes>` elements and any number of `<PackageDefinition>` elements. The `<TransportAttributes>` references complete the other half of the bidirectional referencing scheme that map transport attributes to message definitions.

An example of a `<PackageDefinition>` element can be seen in snippet A in Figure 2. The concept of a Package type includes the concepts of Package structure, Package content, and mapping of content to structure elements. The current implementation of the `<PackageDefinition>` element partially describes the structure and content of the Package. This description includes a mandatory Package Definition Identifier (PDID) and an indicator as to whether or not the described package contains the standard Package header. The `<PackageDefinition>` element is also capable of describing the length of a Package or how to obtain the length of the Package during runtime if the length is variable. Furthermore, the location of the PDID of the Package during runtime can be described if the standard Package header is not being used. A `<PackageDefinition>` element can contain a reference to any number of `<Measurement>` elements, which indicates that a measurement is encapsulated within the described Package.

An example of a `<Measurement>` element can be seen in snippet B in Figure 2. A `<Measurement>` element describes an individual measurement to be acquired during a test. Measurements are related to physical phenomena that occur in the test, and although the model and schema do not yet provide a description of the unit of measurements, they are related to Engineering Units (EU). Currently, the measurement element is a placeholder used to relate named test parameters to the network items that will encapsulate and transport them (Packages and `TmNSDataMessages`). The `<Measurement>` element contains a `<Name>` element that is used for describing the mnemonic of the measurement as well as a `<Description>` element that allows for a textual description of the Measurement.
**Quality of Service Descriptions:** The iNET standards have identified Differentiated Service (DiffServ) as a key technology for implementing quality of service (QoS) in telemetry networks. This technology depends upon markings on packets called differentiated service code points (DSCP). The MDL describes the relationship between TDMs and DSCP values via inclusion of a `<Priority>` element contained in the `<MessageDefinition>` element. The `<Priority>` element references entries in a table that describes the available DSCP settings. The `<DSCPTable>` element contains six default values, two of which can be seen in the example MDL shown in Figure 3.

```
<DSCPTable>
  <DSCPTableEntry>
    <DSCPValue> 101110 </DSCPValue>
    <DSCPDescription> Expected Forwarding </DSCPDescription>
  </DSCPTableEntry>
  <DSCPTableEntry>
    <DSCPValue> 001000 </DSCPValue>
    <DSCPDescription> Assured Forwarding Class 1 </DSCPDescription>
  </DSCPTableEntry>
  ...
</DSCPTable>
```

**Figure 3. MDL Description of DSCP Values**

**Network Descriptions:** A Network, as described by MDL, consists of three major parts:

- NetworkNodes (e.g., DAUs, Recorders, and Switches)
- NetworkConnections
- Networks

A `<Network>` element consists of a set of NetworkNodes, other networks, and the connections between them. Networks can be hierarchical (Networks can contain networks). An example of a TmNS Network, with encapsulated Networks (shown in blue boxes), is shown in Figure 4.
A `<NetworkNode>` element describes a piece of equipment that has an interface to the network. An example segment of an MDL description of a `<NetworkNode>` can be seen in snippet A in Figure 4. The `<RoleID>` element provides a unique identifier that a device manager will utilize to identify which section of the metadata contains configuration information for that `<NetworkNode>`. The `<SNMPConfigInterface>` element describes the configuration object identifiers (OIDs) in the management information bases (MIBs), as well as values for configuring that `<NetworkNode>`. One or more `<NetworkInterface>` elements describe the configuration of the network interfaces, including internet protocol (IP) or media access control (MAC) addresses and physical ports for connecting to `<NetworkNode>`. An example of an MDL description of a `<NetworkInterface>` is provided by snippet B in Figure 4. These elements also contain `<PhysicalNetworkPort>` elements, which describe physical connection points to the `<NetworkNode>`. `<PhysicalNetworkPort>` elements are referenced by `<NetworkConnection>` elements contained in the `<Network>` elements to describe network connectivity. An example of `<NetworkConnection>` XML can be seen in snippet D in Figure 4.

A `<NetworkNode>` element can contain any number of “function” elements. In this example, the `<DAU>` element indicates that the particular `<NetworkNode>` is a Data Acquisition Unit (DAU), which implies that the `<NetworkNode>` supports the TmNS DAU MIB defined by the System Management Standard. The following function elements may be contained by a `<NetworkNode>` element:
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAU</td>
<td>Networked Data Acquisition Unit</td>
</tr>
<tr>
<td>GSE</td>
<td>Ground Support Equipment</td>
</tr>
<tr>
<td>TmnsAdapter</td>
<td>Device implanting TmNS interfaces for two or more devices</td>
</tr>
<tr>
<td>MFD</td>
<td>Multi-Function Display</td>
</tr>
<tr>
<td>Recorder</td>
<td>Network recorder</td>
</tr>
<tr>
<td>SystemManager</td>
<td>Configuration, Control, &amp; Status</td>
</tr>
<tr>
<td>Antenna</td>
<td>Antenna with network control</td>
</tr>
<tr>
<td>Encryptor</td>
<td>Inline network encryptor</td>
</tr>
<tr>
<td>RFNetInterface</td>
<td>Network interface to radio</td>
</tr>
<tr>
<td>RFNetRadio</td>
<td>Implement 2-way wireless network</td>
</tr>
<tr>
<td>Router</td>
<td>Wired network router</td>
</tr>
<tr>
<td>SSTRx</td>
<td>Serial Streaming Telemetry Receiver</td>
</tr>
<tr>
<td>SSTTx</td>
<td>Serial Stream Telemetry Transmitter</td>
</tr>
<tr>
<td>Switch</td>
<td>Wired network switch</td>
</tr>
<tr>
<td>TimeMaster</td>
<td>Time synchronization source</td>
</tr>
</tbody>
</table>

The `<SNMPConfigInterface>` element carries values for configuration variables that will be set by a system manager application via the device’s SNMP interface. This element describes the MIB variables in the SNMP system management interface that are used for configuration, and also provides values for those variables. An instance of an `<SNMPConfigInterface>` element can be used for each MIB for which configuration values are described by the Metadata. `<NetworkNode>` elements may contain any number of `<SNMPConfigInterface>` elements. Figure 5 is an example MDL description of an SNMP Configuration Interface. Many of the OID entries have been removed for brevity.

```
<SNMPConfigInterface>
  
  <OID>.1.3.6.1.4.1.31409</OID>
  <OIDName>TMNS-MIB</OIDName>
  <OIDEntry>
    <RelativeOID>3.7.1</RelativeOID>
    <OIDName>txEnable</OIDName>
    <OLDType>Boolean</OLDType>
    <OIDValue>1</OIDValue>
  </OIDEntry>

  ...
</SNMPConfigInterface>
```

Figure 5. MDL Description of an SNMP Configuration Interface

`<NetworkInterface>` elements contain `<PhysicalNetworkPort>` elements that describe the physical point of connection between the NetworkNodes that make up a Network. Among other optional elements, a `<PhysicalNetworkPort>` contains a `<SpeedKilobitsPerSecond>` element and a `<PortNumber>` element. The `<PortNumber>` element is useful when NetworkNodes have multiple ports, such as in the case of switches and routers. The ID attribute of a
<PhysicalNetworkPort> element is a unique identifier that is referenced by <NetworkConnection> elements to describe the network topology in networks.

CONCLUSION

In conclusion, the iNET project has defined a standard language for describing network-based test systems. The motivation for MDL was to help manage complexity that results in insertion of new technologies and increasing capabilities in the test environment. MDL supports interoperability between devices and applications from different vendors with a standard interchange language.

Telemetry network systems will be configured with MDL instance documents. The syntax of an MDL instance document is enforced by checking against the MDL schema, which is defined using the XML schema language.

MDL describes the networks in a TmNS, including the devices that make up the telemetry network itself and the network topology. Devices include network hardware (switches, routers, encryption hardware, radios, and antennas) and nodes that attach to the network (data acquisition, recording, and data processing devices). Network topology describes the hierarchical breakdown of networks and the interconnections between network interfaces.

MDL also describes the data formats that are published on the network. MDL describes measurements, package definitions, message definitions, transport attributes, and paths. Measurements are encapsulated into packages, and packages are carried in data message payloads. Data messages are published on the network with particular transport attributes. Paths describe sets of connections across which messages are transferred between the message source and sink when a network node subscribes to a particular set of transport attributes. The relationships between these items are described using references between the appropriate XML elements in the instance documents.

The MDL contains numerous descriptive elements that have not been included in this paper. A complete description of MDL will be included in the metadata standard document, which will be published by the MDSWG in proposed form in FY2009. The MDL standard will be further refined by the metadata standards working group as needs arise. The project is executing an assessment and validation of the standard using laboratory systems representative of the anticipated target systems.

ACKNOWLEDGEMENTS

The work described in this paper would not have been possible if not for the contributions from the many members of the iNET Standards Working Group. We gratefully acknowledge this effort and the funding and guidance provided by the iNET program.