

Delay-Tolerant Networking Research Group
Internet-Draft

Intended status: Experimental
Expires: July 4, 2015

E. Birrane
V. Ramachandran
Johns Hopkins Applied Physics Laboratory
December 31, 2014

Delay Tolerant Network Management Protocol
draft-irtf-dtnrg-dtnmp-01

Abstract

This draft describes the Delay/Disruption Tolerant Network Management Protocol (DTNMP). The DTNMP provides monitoring and configuration services between managing devices and managed devices, some of which may operate on the far side of high-delay or high-disruption links. The protocol is designed to minimize the number of transmitted bytes, to operate without sessions or (concurrent) two-way links, and to function autonomously when there is no timely contact with a network operator.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 4, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

Internet-Draft

DTNMP

December 2014

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | | |
|------------------------|--|--------------------|
| 1. | Introduction | 3 |
| 1.1. | Overview | 3 |
| 1.2. | Technical Notes | 4 |
| 1.3. | Scope | 5 |
| 1.3.1. | Protocol Scope | 5 |
| 1.3.2. | Specification Scope | 5 |
| 1.4. | Requirements Language | 5 |
| 2. | Terminology | 5 |
| 3. | System Model | 7 |
| 3.1. | Overview | 7 |
| 3.2. | Roles and Responsibilities | 8 |
| 3.3. | Data Flows | 10 |
| 3.4. | Control Flow by Role | 11 |
| 3.4.1. | Notation | 11 |
| 3.4.2. | Serialized Management | 11 |
| 3.4.3. | Multiplexed Management | 12 |
| 3.4.4. | Data Fusion | 14 |
| 4. | Component Model | 15 |
| 4.1. | Data Representation | 15 |
| 4.1.1. | Types | 15 |
| 4.1.2. | Categories | 15 |
| 4.1.3. | Data Model | 16 |
| 4.2. | Primitive Types | 17 |
| 4.2.1. | Self-Delimiting Numeric Value (SDNV) | 17 |
| 4.2.2. | Timestamp (TS) | 17 |
| 4.2.3. | Data Collections (DC) | 17 |
| 4.3. | Naming and Identification | 17 |
| 4.4. | Special Types | 21 |
| 4.4.1. | MID Collections (MC) | 21 |
| 4.4.2. | Expressions (EXPR) | 21 |
| 4.4.3. | Predicate (PRED) | 22 |
| 5. | Functional Specification | 22 |
| 5.1. | Message Group Format | 23 |
| 5.2. | Message Format | 24 |
| 5.3. | Register Agent (0x00) | 25 |
| 5.4. | Data Report (0x12) | 26 |
| 5.5. | Perform Control (0x20) | 26 |

| | | |
|------------------------|--|--------------------|
| 6. | Appliation Data Model Template | 27 |
| 6.1. | Overview | 27 |
| 6.2. | ADM Types | 27 |
| 6.3. | Template | 28 |
| 6.3.1. | ADM Metadata | 29 |

| | | |
|------------------------|--|--------------------|
| 6.3.2. | ADM Information Capture | 29 |
| 7. | DTNMP Agent Application Data Model | 30 |
| 7.1. | Data Definitions | 30 |
| 7.2. | Metadata Definitions | 31 |
| 7.3. | Data Definitions | 31 |
| 7.3.1. | Atomic Data | 31 |
| 7.3.2. | Computed Data | 32 |
| 7.3.3. | Reports | 32 |
| 7.4. | Operators | 33 |
| 7.5. | Controls | 35 |
| 7.5.1. | Control Summary | 35 |
| 7.5.2. | Control Specification | 37 |
| 8. | IANA Considerations | 55 |
| 9. | Security Considerations | 55 |
| 10. | Normative References | 55 |
| | Authors' Addresses | 55 |

[1.](#) Introduction

This document presents the Delay/Disruption Tolerant Network Management Protocol (DTNMP) providing application-layer network management services over links where propagation delays or link disruptions prevent timely communications between a network operator and a managed device.[\[RFC4838\]](#).

[1.1.](#) Overview

A network management protocol defines the messages that implement management functions amongst managed and managing devices in a network. Management functions include the definition, production, and reporting of performance data, the application of administrative and security policy, and the configuration of behavior based on local time and state measurements.

DTNs contain nodes whose communication links are characterized by signal propagation delays and/or transmission disruptions that make

timely data exchange difficult or impossible. Management protocols that rely on timely data exchange, such as those that rely on negotiated sessions or other synchronized acknowledgment, do not function in the DTN environment. For example, the Internet approach of network management via closed-loop, synchronous messaging does not work when network disruptions increase in frequency and severity. While no protocol delivers data in the absence of a networking link, protocols that eliminate or drastically reduce overhead and end-point coordination require smaller transmission windows and continue to function when confronted with scaling delays and disruptions in the network.

DTNMP accomplishes the network management function using open-loop, intelligent-push, asynchronous mechanisms that better scale as link challenges scale. The protocol is designed to support five desirable properties:

Intelligent Push of Information

The intelligent push of information eliminates the need for round-trip data exchange in the management protocol. This is a necessary consequence of operating in open-loop systems where reliable round-trip communication may not exist. DTNMP is designed to operate even in uni-directional networks.

Small Message Sizes

Smaller messages require smaller periods of viable transmission for communication, incur less re-transmission cost, and consume less resources when persistently stored en-route in the network. DTNMP minimizes the size of a message whenever practical, to include packing and unpacking binary data, variable-length fields, and pre-configured data definitions.

Fine-grained, Flexible Data Identification

Fine-grained identification allows data in the system to be explicitly addressed while flexible data identification allows users to define their own customized, addressed data collections. In both cases, the ability to define precisely the data required removes the need to query and transmit large data sets only to filter/downselect desired data at a receiving device.

Asynchronous Operation

DTNMP does not rely on session establishment or round-trip data exchange to perform network management functions. Wherever possible, the DTNMP is designed to be stateless. Where state is required, the DTNMP provides mechanisms to support transactions and graceful degradation when nodes in the network fail to synchronize on common definitions.

Compatibility with Low-Latency Network Management Protocols

DTNMP adopts an identifier approach compatible with the Managed Information Base (MIB) format used by Internet management protocols such as the Simple Network Management Protocol (SNMP), thus enabling management interfaces between DTNs and other types of networks.

[1.2.](#) Technical Notes

All multi-byte values are assumed to be communicated in network-byte order. Bit-fields are specified in Little-Endian format with bit

position 0 holding the least-significant bit (LSB). When illustrated in this manuscript, the LSB appears on the right.

[1.3.](#) Scope

[1.3.1.](#) Protocol Scope

The DTNMP provides data monitoring, administration, and configuration for applications operating above the data link layer of the OSI networking model. While the DTNMP may be configured to support the management of network layer protocols (such as the Internet Protocol and Bundle Protocol) it also uses these protocols stacks to encapsulate and communicate its own DTNMP messages.

It is assumed that the protocols used to carry DTNMP messages provide addressing, confidentiality, integrity, security, fragmentation support and other network/session layer functions. Therefore, these items are not discussed in the scope of this document.

[1.3.2.](#) Specification Scope

This document describes the format of the DTNMP messages exchanged

amongst managing and managed devices in a DTN. This document further describes the rationale behind key design decisions to the extent that such a description informs the operational deployment and configuration of DTNMP implementations. This document does not address specific data configurations of DTNMP-enabled devices, nor does it discuss the interface between DTNMP and other management protocols, such as SNMP.

[1.4.](#) Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

[2.](#) Terminology

This section identifies those terms critical to understanding the proper operation of the DTNMP. Whenever possible, these terms align in both word selection and meaning with their analogs from other management protocols.

Actor

A software service running on either managed or managing devices implementing an end-point in the DTNMP. Actors may implement the "Manager" role, "Agent" role, or both.

Agent Role

A role within the DTNMP, associated with a managed device, responsible for reporting performance data, enforcing administrative policies, and accepting/performing actions. Agents exchange information with DTNMP managers operating either on the same device or on a remote managing device.

Application Data Model (ADM)

The set of predefined data definitions, reports, literals, operations, and controls given to a DTNMP actor to manage a particular application or protocol. DTNMP actors support multiple ADMs, one for each application/protocol being managed.

Atomic Data

Globally unique, managed data definitions, similar to those defined in a Management Information Base (MIB), whose definition does not change based on the configuration of a DTNMP actor. Atomic data comprise the "lingua franca" within the DTNMP. All messages in the protocol operate either directly on atomic data or on data derived from atomic data.

Computed Data

Data items that are computed dynamically by a DTNMP actor from some combination of Atomic Data and other Computed Data.

Controls

Operations that may be undertaken by a DTNMP actor to change the behavior, configuration, or state of an application managed by the DTNMP.

Macros

A named, ordered collection of controls.

Managed Item Definition (MID)

A parameterized structure used to uniquely identify all data and control definitions within the DTNMP. MIDs are a superset of Object Identifiers (OIDs) and the mechanism by which the DTNMP maintains data compatibility with other management protocols.

Manager

A role within the DTNMP associated with a managing device responsible for configuring the behavior of, and receiving/processing/visualizing information from, DTNMP agents. DTNMP managers also provide gateways to non-DTNMP management protocols as part of conditioning the data returned from agents. Managers interact with one or more

agents located on the same device and/or on remote devices in the network.

Report

A named, ordered collection of data items gathered by one or more DTNMP agents and provided to one or more DTNMP managers. Reports may contain atomic data, computed data, and other reports. Individual data within a report are not named; the

report itself is named to reduce the size of the report message.

[3.](#) System Model

[3.1.](#) Overview

DTNMP performs the core network management functions of configuration, performance reporting, control, and administration, as follows.

Configuration

The configuration function synchronizes definitions amongst DTNMP actors in the DTN. Managers and Agents must agree on what ADMs are supported by what nodes. Further, these Actors must agree on the definitions of customized information, such as data production schedules, report definitions, and state-based autonomous actions.

Performance Reporting

Since DTNMP operates asynchronously, performance **monitoring** is replaced by performance **reporting**. As there is no expectation of closed-loop control of a managed device across a delayed/disrupted link, the best action that a managing device can undertake is to collect and operate on whatever data is received by managed devices.

Reporting the performance of a managed device involves the local collection of reports and the communication of those reports to appropriate managing devices.

Control

Part of the ADM for a supported application/protocol includes a list of controls/commands that may be run by a DTNMP actor based on local time or local state. Controls comprise the basic autonomy mechanism within the DTNMP.

Administration

The mappings amongst agents and managers within a network may be complex, especially in networks comprising multiple administrative domains. The administrative management

function defines what managers may control what agents, for

what types of information.

3.2. Roles and Responsibilities

By definition, DTNMP agents reside on managed devices and DTNMP managers reside on managing devices. These roles naturally map to the transmit and receipt of various DTNMP messages. This section describes how each of these roles participate in the network management functions outlined in the prior section.

Agent Responsibilities

Local Data Collection

Agents **MUST** collect and report the data defined in all ADMs for which they have been configured for the local managed device. Agents **MAY** also collect data for network nodes that do not have their own DTNMP agents. In this scenario, the DTNMP agent acts as a proxy agent.

Autonomous Control

Agents **MUST** determine, without manager intervention, whether a configured control should be invoked. Agents **MUST** periodically evaluate the conditions associated with configured controls and invoke those controls based on local state. Agents **MAY** also invoke controls on other devices within a regional, low-latency network.

Data Conditioning

DTNMP agents **MUST** accept computed data definitions that specify how a single data value may be constructed from the transformation of one or more other data values in the system, using the expression syntax specified in this manuscript. Further, agents **MUST** produce this data when requested by Managers with the appropriate security permissions. Agents **MUST** produce the list of computer data definitions when requested by a Manager. Agents **MUST** detect when a computed data definition references other data definitions that are unknown to the agent and respond in a way consistent with the logging/error-handling configuration of the agent.

Report Definition

Agents **MUST** support the ability to accept and store definitions for custom report definitions. Agents

MUST conform to the security policies associated with custom reports when determining if a Manager may request a report defined by a different Manager in the network. Agents MUST provide a listing of custom report definitions to appropriate managing devices when requested. Agents MUST detect requests for custom reports that are not configured on the agent, or are not appropriate for the requesting Manager, and respond in a way consistent with the logging/error-handling configuration of the agent.

Consolidate Messages

Agents SHOULD produce as few messages as possible when sending information. For example, rather than sending multiple report messages to a manager, an agent SHOULD prefer to send a single message containing multiple reports.

Regional Proxy

Agents MAY perform any of their responsibilities on behalf of other network nodes that, themselves, do not have a DTNMP agent. In such a configuration, the DTNMP agent acts as a proxy agent for these other network nodes.

Manager Responsibilities

Agent/ADM Mapping

Managers MUST understand what ADMs are supported by the various agents with which they communicate. Managers SHOULD NOT attempt to request, invoke, or refer to ADM information for ADMs unsupported by an agent.

Data Collection

Managers MUST receive information from agents by asynchronously configuring the production of data reports and then waiting for, and collecting, responses from agents over time. Managers SHOULD implement internal time-outs to detect conditions where agent information has not been received within network-specific timespans.

Custom Definitions

Managers SHOULD provide the ability to define custom data and report definitions. Any defined custom

definitions MUST be transmitted to appropriate agents
and these definitions MUST be remembered to interpret

the reporting of these custom values from an agent in
the future.

Data Translation

Managers SHOULD provide some interface to other
network management protocols, such as the SNMP.
Managers MAY accomplish this by accumulating a
repository of push-data from high-latency parts of
the network from which data may be pulled by low-
latency parts of the network.

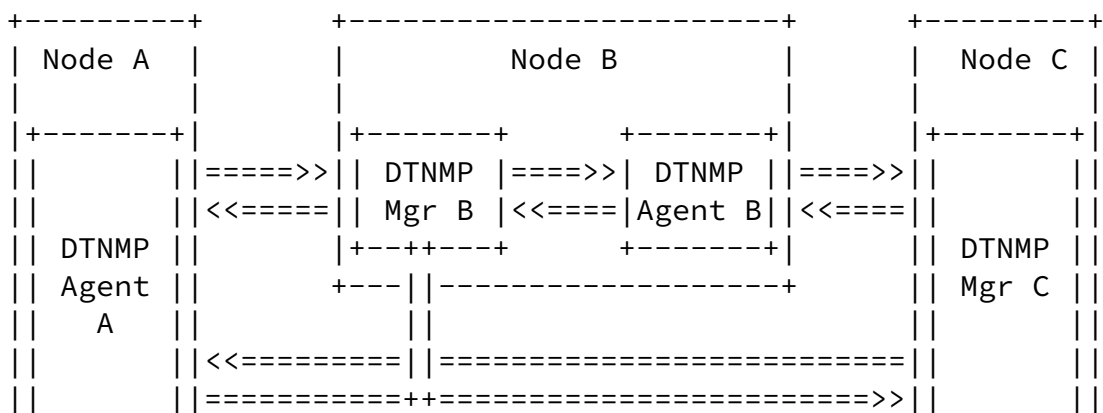
Data Fusion

Managers MAY support the fusion of data from multiple
agents with the purpose of transmitting fused data
results to other managers within the network.
Managers MAY receive fused reports from other
managers pursuant to appropriate security and
administrative configurations.

[3.3.](#) Data Flows

We identify three significant data flows within the DTNMP: control
flows from managers to agents, reports flows from agents to managers,
and fusion reports from managers to other managers. These data flows
are illustrated in Figure 1.

DTNMP Data Flows



```
|+-----+|
+-----+
```

```
|+-----+|
+-----+
```

Figure 1

In this data flow, the agent on node A receives configurations from managers on nodes B and C, and replies with reports back to these managers. Similarly, the agent on node B interacts with the local manager on node B and the remote manager on node C. Finally, the

manager on node B may fuse data reports received from agents at nodes A and B and send these fused reports back to the manager on node C. From this figure, we see many-to-many relationships amongst managers, amongst agents, and between agents and managers. Note that agents and managers are roles, not necessarily differing software applications. Node A may represent a single software application fulfilling only the agent role, whereas node B may have a single software application fulfilling both the agent and manager roles. The specifics of how these roles are realized is an implementation matter.

3.4. Control Flow by Role

This section describes three common configurations of DTNMP agents and managers and the flow of messages between them. These configurations involve local and remote management and data fusion.

3.4.1. Notation

The notation outlined in Table 1 describes the types of control messages exchanged between DTNMP agents and managers.

| Term | Definition | Example |
|---------------------|--|---------------------|
| AD# | Atomic data definition, from ADM. | AD1 |
| CD# | Custom data definition. | CD1 = AD1 + CD0. |
| DEF([ACL], ID,EXPR) | Define id from expression. Allow managers in access control list (ACL) | DEF([*], CD1, AD1 + |

| | | |
|------------|---|------------------|
| | to request this id. | AD2) |
| PROD(P,ID) | Produce ID according to predicate P. P may be a time period (1s) or an expression (AD1 > 10). | PROD(1s, AD1) |
| RPT(ID) | A report identified by ID. | RPT(AD1) |

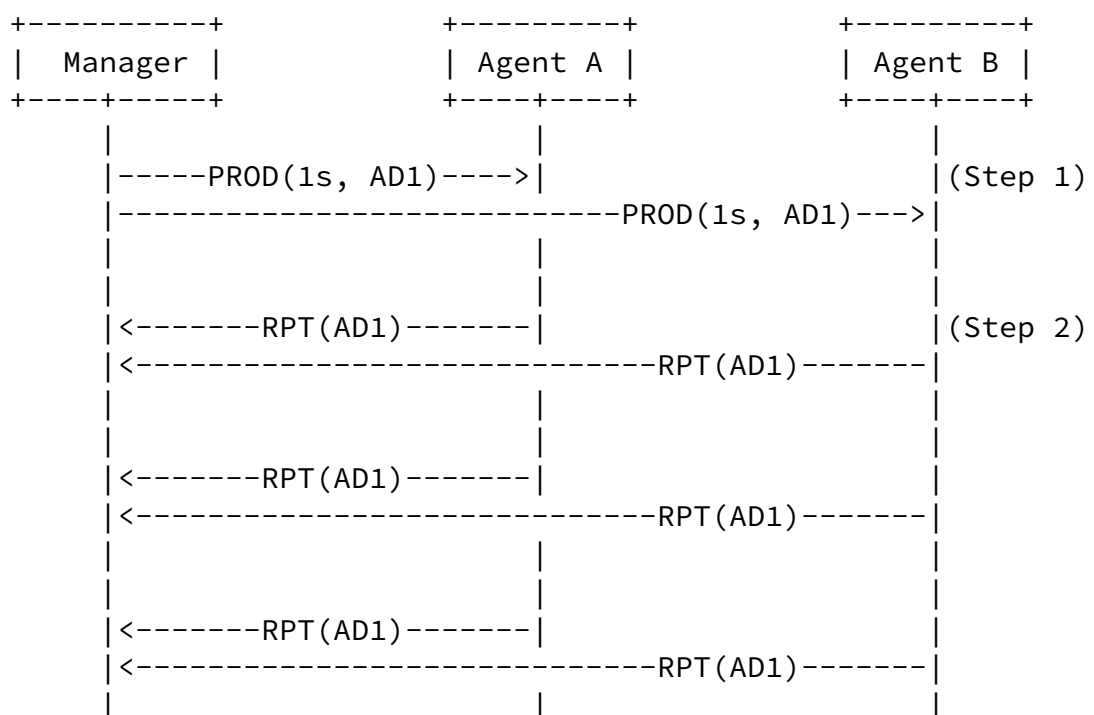
Table 1: Terminology

3.4.2. Serialized Management

This is a nominal configuration of network management where a managing device interacts with a set of managed devices, with a DTNMP manager installed on the managing device and a DTNMP agent installed

on each managed device. The control flows for this are outlined in Figure 2.

Serialized Management Control Flow



In a simple network, a manager interacts with multiple agents.

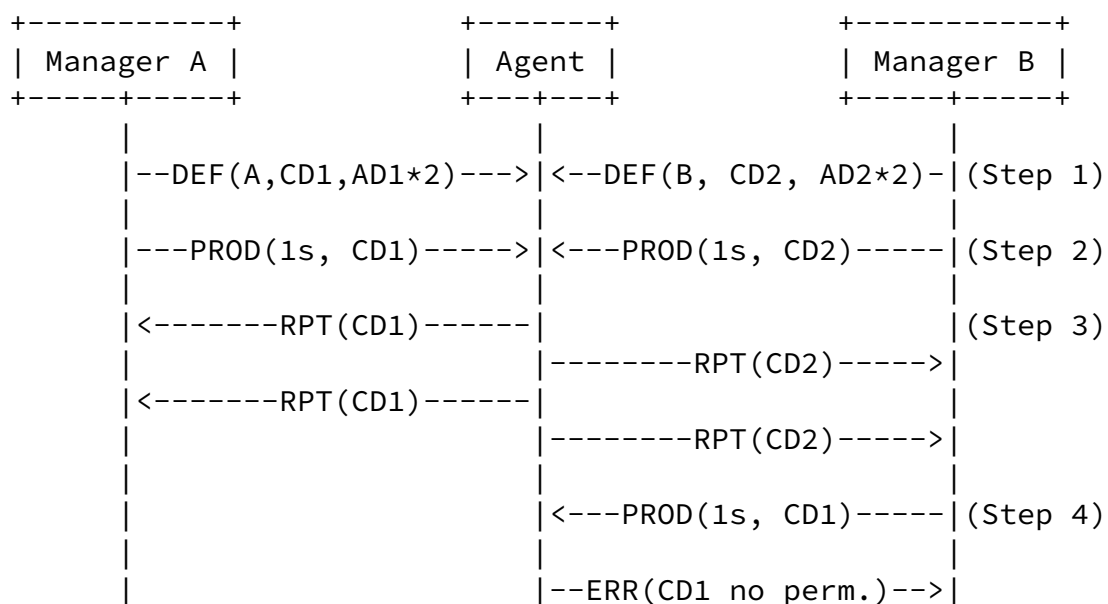
Figure 2

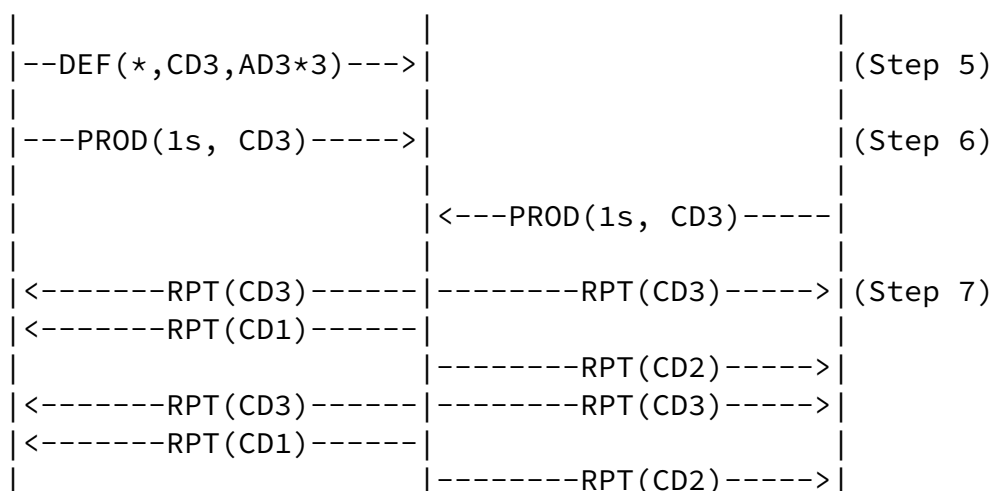
In this figure, the manager configures agents A and B to produce atomic data AD1 every second in (Step 1). At some point in the future, upon receiving and configuring this message, agents A and B then build a report containing AD1 and send those reports back to the manager in (Step 2).

3.4.3. Multiplexed Management

Networks spanning multiple administrative domains may require multiple managing devices (for example, one per domain). When a manager defines custom reports/data to an agent, that definition may be tagged with an access control list (ACL) to limit what other managers will be privy to this information. Managers in such networks SHOULD synchronize with those other managers granted access to their custom data definitions. When agents generate messages, they MUST only send messages to managers according to these ACLs, if present. The control flows in this scenario are outlined in Figure 3.

Multiplexed Management Control Flow





Complex networks require multiple managers interfacing with agents.

Figure 3

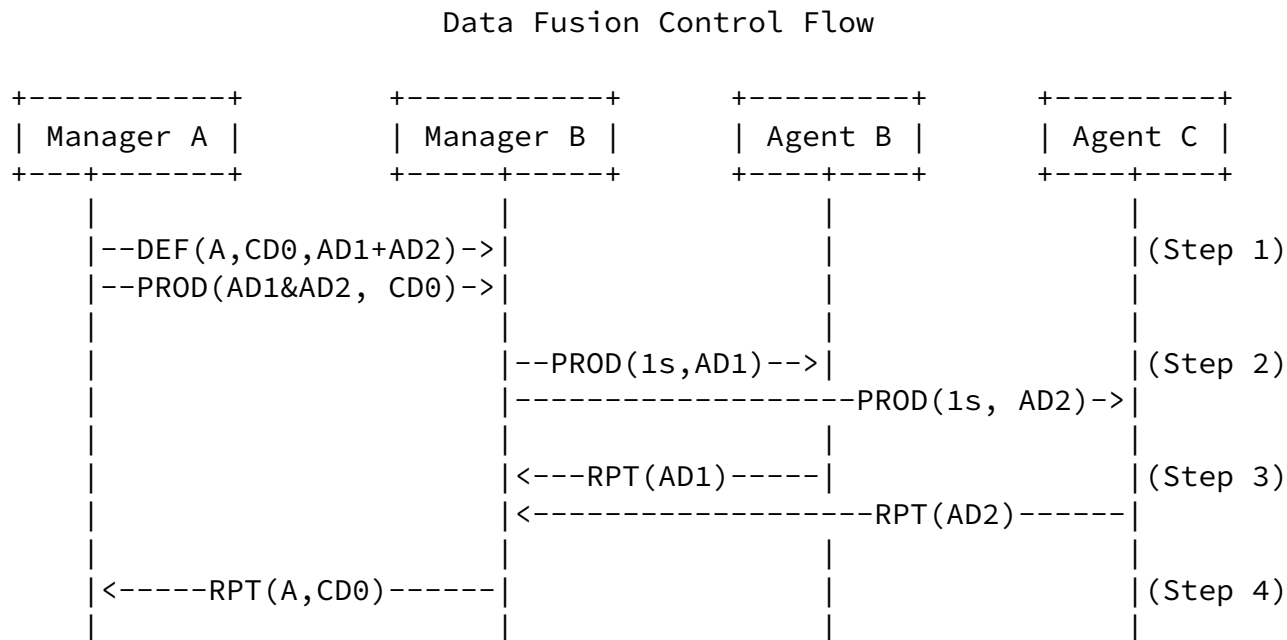
In more complex networks, managers may choose to define custom reports and data definitions, and agents may need to accept such definitions from multiple managers. Custom data definitions may include an ACL that describes who may query and otherwise understand the custom definition. In (Step 1), Manager A defines CD1 only for A while Manager B defines CD2 only for B. Managers may, then, request the production of reports containing these custom definitions, as shown in (Step 2). Agents produce different data for different managers in accordance with configured production rules, as shown in (Step 3). If a manager requests an operation, such as a production rule, for a custom data definition for which the manager has no permissions, a response consistent with the configured logging policy

on the agent should be implemented, as shown in (Step 4). Alternatively, as shown in (Step 5), a manager may define custom data with no restrictions allowing all other managers to request and use this definition. This allows all managers to request the production of reports containing this definition, shown in (Step 6) and have all managers receive this and other data going forward, as shown in (Step 7).

[3.4.4.](#) Data Fusion

In many networks, agents do not want to individually transmit their

data to a manager, preferring instead to fuse reporting data with local nodes prior to transmission. This approach reduces the number and size of messages in the network and reduces overall transmission energy expenditure. DTNMP supports fusion of NM reports by co-locating agents and managers on managed devices and offloading fusion activities to the manager. This process is illustrated in Figure 4.



Data fusion in DTNMP accours amongst managers in the network.

Figure 4

In this example, Manager A requires the production of a computed data set, CD0, from node B, as shown in (Step 1). The manager role understands what data is available from what agents in the subnetwork local to B, understanding that AD1 is available locally and AD2 is available remotely. Production messages are produced in (Step 2) and data collected in (Step 3). This allows the manager at node B to fuse the collected data reports into CD0 and return it in (Step 4). While a trivial example, the mechanism of associating fusion with the

manager function rather than the agent function scales with fusion complexity, though it is important to reiterate that agent and manager designations are roles, not individual software components. There may be a single software application running on node B

implementing both Manager B and Agent B roles.

[4. Component Model](#)

This section identifies the components that comprise the data communicated within the DTNMP, the way in which these components are named, and those special types associated with DTNMP messages.

[4.1. Data Representation](#)

[4.1.1. Types](#)

Components within the DTNMP are represented as one of four basic data types: Data, Controls, Literals, and Operators:

Data Data consist of values collected by an agent and reported to managers within the DTNMP. This includes definitions from an ADM, derived data as configured from managers, and reports which are collections of data elements.

Controls Controls consist of actions that may be invoked on agents and managers to change behavior in response to some external event (such as local state changes or time). Controls include application-specific functions specified as part of an ADM and macros which are collections of these controls.

Literals Literals are constant numerical values that may be used in the evaluation of expressions and predicates.

Operator Operators are those mathematical functions that operate on series of Data and Literals, such as addition, subtraction, multiplication, and division.

[4.1.2. Categories](#)

Components within the DTNMP can be categorized as Atomic, Computed, or Collection.

Atomic

Atomic components are those components that are directly implemented by the underlying software/firmware of a network node. Atomic components may also refer to components whose definitions may not be changed. Examples of atomic components are Data, Controls, Literals, and Operators

specified by an ADM. Atomic component identifiers **MUST** be unique and **SHOULD** be managed by a registration authority, similar to the mechanisms used to assign Object Identifiers (OIDs). Atomic components must be understood by both DTNMP managers and agents in a network.

Computed

Computed components are those components that are not directly implemented by the underlying software/firmware of a network node. The definition of a computed component **MAY** be dynamically defined by DTNMP managers and communicated to one or more DTNMP agents in a network. The definition of a computed component may include other computed components or other atomic components. The identifier of a computed component is not guaranteed to be universally unique but **SHOULD** be unique within the context of a particular network or internetwork.

Collection

A collection component is a set of other components (to include other collection components). DTNMP implementations **MUST** prevent circular definitions when implementing collections that include other collections.

[4.1.3.](#) Data Model

Each component of the DTNMP data model can be identified as a combination of type and category, as illustrated in Table 2. In this table type/category combinations that are unsupported are listed as N/A. Specifically, DTNMP does not support user-defined controls, constants, or operations; any value that specifies action on an agent **MUST** be pre-configured as part of an ADM.

| | Data | Controls | Literals | Operator |
|------------|------------------|----------|----------|----------|
| Atomic | Measured Value | Control | Constant | Operator |
| Computed | Calculated Value | N/A | N/A | N/A |
| Collection | Report | Macro | N/A | N/A |

Table 2

Internet-Draft

DTNMP

December 2014

[4.2.](#) Primitive Types

[4.2.1.](#) Self-Delimiting Numeric Value (SDNV)

The data type "SDNV" refers to a Self-Delimiting Numerical Value (SDNV) described in [\[RFC6256\]](#). SDNVs are used in the DTNMP to capture any data items that are expected to be 8 bytes or less in total length. DTNMP actors MAY reject any SDNV value that is greater than 8 bytes in length.

[4.2.2.](#) Timestamp (TS)

For compatibility with a variety of protocols, the use of UTC time is selected to represent all time values in the DTNMP. However, timestamps in DTNMP may represent either absolute or relative time based on the associated epoch. DTNMP uses September 9th, 2012 as the timestamp epoch (UTC time 1348025776). Values less than this value MUST be considered as relative times. Values greater than or equal to this epoch MUST be considered as absolute times. In all cases, the DTNMP timestamp is encoded as an SDNV.

[4.2.3.](#) Data Collections (DC)

A Data collection is comprised of a value identifying the number of bytes in the collection, followed by each byte, as illustrated in Figure 5. Data collections are used in the DTNMP to capture variable data sets that are too large to place in an SDNV.

Data Collection

```

+-----+-----+-----+      +-----+
| # Bytes | BYTE 1 | BYTE 2 | ... | BYTE N |
| [SDNV]  | [BYTE] | [BYTE] |   | [BYTE] |
+-----+-----+-----+      +-----+
```

Figure 5

[4.3.](#) Naming and Identification

While all components within the DTNMP can be uniquely identified by

an Object Identifier (OID), several annotations exist to assist with the filtering and access control associated with components beyond what is efficiently represented in the OID structure. These annotations include the type and category of the component, an optional identifier naming the issuer of the component, and an optional tag value holding issuer-specific information about the component.

The DTNMP Managed Identifier (MID) provides a single, variable length structure that encapsulates all of the information necessary to identify a DTNMP component and its useful annotations.

The MID structure, illustrated in Figure 6, is comprised of up to four fields. In this illustration, each field is named, the type of each field is given in []'s, and the string "(opt.)" indicates that the field is optional, pending on the values found in the flags byte.

MID format

| | | | |
|---------|---------|----------|---------|
| +-----+ | +-----+ | +-----+ | +-----+ |
| Flags | Issuer | OID | Tag |
| [BYTE] | [SDNV] | [VARIED] | [SDNV] |
| | (opt.) | | (opt.) |
| +-----+ | +-----+ | +-----+ | +-----+ |

Figure 6

The MID fields are defined as follows.

Flags

Flags are used to describe the type of component identified by the MID, identify which optional fields in the MID are present, and the encoding used to capture the component's OID. The layout of the flag byte is illustrated in Figure 7.

MID Flag Format

| | | | |
|---------|---------|---------|------------|
| +-----+ | +-----+ | +-----+ | +-----+ |
| OID | TAG | ISS | CAT TYPE |
| +-----+ | +-----+ | +-----+ | +-----+ |
| 7 6 | 5 | 4 | 3 2 1 0 |

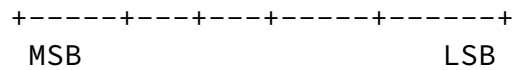


Figure 7

MID Type (TYPE)

The type of the component encapsulated by the MID, enumerated as data (0), control (1), literal (2), or operator (3).

MID Category (CAT)

The category of the component encapsulated by the MID, enumerated as atomic (0), computed (1), and collection (2).

Issuer Present (ISS)

Whether the issuer field is present (1) or not (0) for this MID. If this flag has a value of 1 then the issuer field MUST be present in the MID. Otherwise, the issuer field MUST NOT be present in the MID.

Tag Present (TAG)

Whether the tag field is present (1) or not (0) for this MID. If this flag has a value of 1 then the tag field MUST be present in the MID. Otherwise, the tag field MUST NOT be present.

OID Type (OID)

Whether the contained OID field represents an full OID (0), a parameterized OID (1), a compressed full OID (2), or a compressed, parameterized OID (3).

For example, a MID flag byte of 0x00 indicates an atomic data value with no issuer or tag field encapsulating a full OID. A MID flag byte of 0x94 indicates a computed data value with an issuer field, but no tag field encapsulating a compressed OID.

Issuer

This is a binary identifier representing a predetermined issuer name. The DTNMP protocol does not parse or validate

this identifier, using it only as a distinguishing bit pattern to assure MID uniqueness. This value, for example, may come from a global registry of organizations, an issuing node address, or some other network-unique marking.

OID

The core of a MID is its encapsulated Object Identifier (OID). Aside from the flag byte, this is the only other mandatory element within a MID. The DTNMP defines four types of OID references for this part of the MID structure: Full OIDs, Parameterized OIDs, Compressed Full OIDs, and Compressed Parameterized OIDs.

Full OID

This is a binary representation of the full OID associated with the named value. The OID is encoded using a modified form of the ASN.1 Basic Encoding Rules (BER) for Object Identifiers (type value of 0x06). In the standard ASN.1 encoding, four octet sets are defined: identifier octets, length octets, contents octets, and end-of-contents octets. A DTNMP Full OID does not use the identifier, length, or end-

of-contents octets. Instead, a DTNMP Full OID is comprised of two fields: the length in bytes of the encoded OID captured in an SDNV followed by the OID contents octets, as illustrated in Figure 8.

Full OID Format

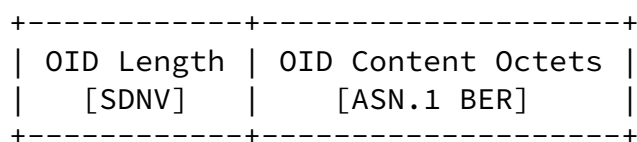


Figure 8

Parameterized OID

The parameterized OID is represented as the non-parameterized portions of the OID followed by one or more parameters. Parameterized OIDs are used to templatzize the specification of data items and

otherwise provide parameters to controls without requiring potentially unmanagable growth of a Full OID namespace. The format of a parameterized OID is given in Figure 9.

Parameterized OID Format

| | | | | |
|----------|---------|---------|-----|---------|
| +-----+ | +-----+ | +-----+ | | +-----+ |
| Base OID | # Parms | Parm 1 | ... | Parm N |
| [VAR] | [SDNV] | [DC] | | [DC] |
| +-----+ | +-----+ | +-----+ | | +-----+ |

Figure 9

Compressed Full OID

Since many related OIDs share a common and lengthy hierarchy there is opportunity for significant message size savings by defining a shorthand for commonly-used portions of the OID tree. A partial OID is a tuple consisting of a nickname for a pre-defined portion of the OID tree (as an SDNV), followed by a relative OID.

Compressed Parameterized OID

A compressed, parameterized OID is similar to a compressed OID. In this instance, the tuple contained in this field is the nickname for the pre-defined portion of the OID tree (as an SDNV) followed

by a parameterized OID whose hierarchy begins at the place identified by the nickname.

Tag

A value used to disambiguate multiple MIDs with the same OID/Issuer combination. The definition of the tag is left to the discretion of the MID issuer. Proper name objects do not require a tag as their OIDs are guaranteed to be globally unique. Options for tag values include an issuer-known version number or a hashing of the data associated with a non-proper-name MIDs. The tag field MUST NOT be present for the atomic category.

[4.4.](#) Special Types

In addition to the primitive data types already mentioned, the following special data types are also defined.

[4.4.1.](#) MID Collections (MC)

A MID collection is comprised of a value identifying the number of MIDs in the collection, followed by each MID, as illustrated in Figure 10.

MID Collection

| | | | |
|---------------|-------|---------|-------|
| +-----+-----+ | | +-----+ | |
| # MIDs | MID 1 | ... | MID N |
| [SDNV] | [MID] | | [MID] |
| +-----+-----+ | | +-----+ | |

Figure 10

[4.4.2.](#) Expressions (EXPR)

Expressions apply operations to data and literal values to generate new data values. The expression type in DTNMP is a collection of MIDs that represent a postfix notation stack of Data, Literal, and Operation types. For example, the infix expression $A * (B * C)$ is represented as the sequence $A B C * *$. The format of an expression is illustrated in Figure 11.

Expression Format

| | |
|---------------|------------|
| +-----+-----+ | |
| Priority | Expression |
| [SDNV] | [MC] |

Figure 11

Priority

The priority of this expression relative to any other expression configured on the DTNMP actor. Priorities are used when one expression MUST be evaluated before some other expression is evaluated. This field represents an unsigned integer value with larger values indicating higher priority. Unless otherwise specified, a default priority value of 0 SHALL be used for any defined expression.

Expression

An expression is represented in the DTNMP as a MID collection, where each MID in the ordered collection represents the data, literals, and operations that comprise the expression.

[4.4.3.](#) Predicate (PRED)

Predicates are expressions whose values are interpreted as a Boolean. The value of zero MUST be considered "false" and all other values MUST be considered "true". Similar to an expression, a predicate is represented as a MID collection.

[5.](#) Functional Specification

This section describes the format of the messages that comprise the DTNMP protocol. When discussing the format/types of data that comprise message fields, the following conventions are used.

| Type Name | Description |
|-----------|---------------------------------|
| BYTE | Unsigned, 8-bit byte. |
| DC | Data Collection |
| EXPR | Expression |
| MC | MID Collection |
| MID | Managed Identifier |
| PRED | Predicate |
| SDNV | Self-Delimiting Numerical Value |
| TS | Timestamp |
| VAR | Variable field. |

5.1. Message Group Format

Individual messages within the DTNMP are combined into a single group for communication with another DTNMP actor. Messages within a group **MUST** be received and applied as an atomic unit. The format of a message group is illustrated in Figure 12. These message groups are assumed communicated amongst agents and managers as the payloads of encapsulating protocols, such as the Bundle Protocol or Internet Protocol, which **MAY** provide additional security and data integrity features.

DTNMP Message Group Format

| # Msgs | Timestamp | Message 1 | ... | Message N |
|--------|-----------|-----------|-----|-----------|
| [SDNV] | [TS] | [VAR] | | [VAR] |

Figure 12

Msgs

The number of messages that are together in this message group.

Timestamp

Internet-Draft

DTNMP

December 2014

The creation time for this messaging group. This timestamp MUST be an absolute time. Individual messages may have their own creation timestamps based on their type, but the group timestamp also serves as the default creation timestamp for every message in the group.

Message N

The Nth message in the group.

5.2. Message Format

Each message identified in the DTNMP specification adheres to a common message format, illustrated in Figure 13, consisting of a message header, a message body, and an optional trailer.

DTNMP Message Format

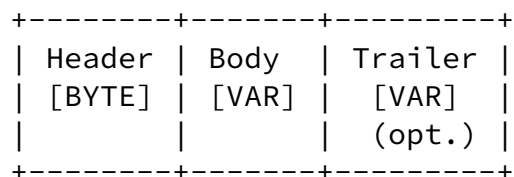


Figure 13

Header

The message header byte is shown in Figure 14. The header identifies a message context and opcode as well as flags that control whether a report should be generated on message success (Ack) and whether a report should be generated on message failure (Nack).

DTNMP Common Message Header

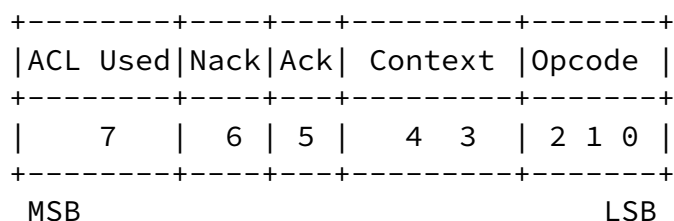


Figure 14

Opcode

The opcode field identifies the opcode of the message, within the associated message context.

ACK Flag

The ACK flag describes whether successful application of the message must generate an acknowledgement back to the message sender. If this flag is set (1) then the receiving actor **MUST** generate a report communicating this status. Otherwise, the actor **MAY** generate such a report based on other criteria.

NACK Flag

The NACK flag describes whether a failure applying the message must generate an error notice back to the message sender. If this flag is set (1) then the receiving actor **MUST** generate a report communicating this status. Otherwise, the actor **MAY** generate such a report based on other criteria.

ACL Used Flag

The ACL used flag indicates whether the message has a trailer associated with it that specifies the list of DTNMP actors that may participate in the actions or definitions associated with the message. This area is still under development.

Body

The message body contains the information associated with the given message.

Trailer

An **OPTIONAL** access control list (ACL) may be appended as a trailer to a message. When present, the ACL for a message identifies the agents and managers that can be affected by the definitions and actions contained within the message. The explicit impact of an ACL is described in the context of each message below. When an ACL trailer is not present, the message results may be visible to any DTNMP actor in the

network, pursuant to other security protocol implementations.

[5.3.](#) Register Agent (0x00)

The Register Agent message is used to inform a DTNMP manager of the presence of another agent in the network.

```
+-----+
| Agent ID |
| [SDNV]   |
+-----+
```

Figure 15: Register Agent Message Body

Agent ID

The Agent ID MUST represent the unique address of the agent in whatever protocol is used to communicate with the agent. For example, when DTNMP is run over Bundle Protocol, the Agent ID should be the Endpoint Identifier (EID) of the agent being added.

[5.4.](#) Data Report (0x12)

Data reports include a listing of one or more data items collected from a managed device. These reports may include atomic data, computed data, or any report definition known to the generating device. Each message is a concatenation of ID/Data definitions with the overall message length assumed to be captured in the underlying transport container.

```
+-----+-----+-----+-----+      +-----+-----+
| Time | Num | ID 1 | Data 1 |      | ID N | Data N |
| [TS] |[SDNV]| [MID]| [DC] |...| [MID]| [DC] |
+-----+-----+-----+-----+      +-----+-----+
```

Figure 16

Time

The time at which the report was generated by the DTNMP actor.

Num

The number of reports in the data report message.

ID N

The MID identifying the Nth report.

Data N

The contents of the Nth report.

[5.5.](#) Perform Control (0x20)

The perform control method causes the receiving DTNMP actor to apply one or more pre-configured controls.

Predicate Production Message

```
+-----+-----+
| Start | Controls |
| [TS]  | [MC]    |
+-----+-----+
```

Figure 17

Start

The time at which the control should be run.

Controls

The collection of controls to be applied by the DTNMP actor. The MID identifying a control will contain the parameters for the control (if any) through the use of a parameterized OID captured within the MID.

[6.](#) Appliation Data Model Template

[6.1.](#) Overview

An application data model (ADM) specifies the set of DTNMP components associated with a particular application/protocol. The purpose of the ADM is to provide a guaranteed interface for the management of an application or protocol over DTNMP that is independent of the nuances of its software implementation. In this respect, the ADM is conceptually similar to the Managed Information Base (MIB) used by SNMP, but contains additional information relating to command opcodes and more expressive syntax for automated behavior.

Currently, the ADM is an organizing document and not used to automatically generate software. As such, the ADM template lists the kind of information that must be present in an ADM definition but does not address mechanisms for automating implementations.

Each ADM specifies the globally unique identifiers and descriptions for all data, controls, literals, and operators associated with the application or protocol managed by the ADM. Any implementation claiming compliance with a given ADM must compute all identified data, perform identified controls, and understand identified literals and operators.

6.2. ADM Types

ADMs must specify data types when defining data items and parameters. In addition to the standard DTNMP Types (SDNV, TS, DC, MID, OID,

P_OID, MC, EXPR, and PRED) ADMs support the following additional data types.

| Data Type | ID | Description | Encapsulating DTNMP Type |
|-----------|-----|--|--------------------------|
| Integer | INT | A signed or unsigned integer up to 64 bits in width encoded using Google Protocol Buffer VARINT rules. Booleans, enumerations, and all integer values are captured | SDNV |

| | | | | |
|---------------------|--------|---|-----------------------|--|
| | | | using this data type. | |
| Float | FLOAT | A 32-bit fixed-width number stored according to the IEEE-754 float standard. | SDNV | |
| Double | DOUBLE | A 64-bit fixed-width number stored according to the IEEE-754 double standard. | SDNV | |
| STRING | STR | An array of characters preceded by the character count. Strings are not required to be NULL-terminated. | DC | |
| Binary Large Object | BLOB | An undefined series of user-defined bytes preceded by the length of the binary object. This data type is captured . | DC | |

Table 3: ADM Data Types

6.3. Template

ADM definitions specify the metadata, data, controls, literals, and operators associated with a managed application or protocol.

6.3.1. ADM Metadata

ADM metadata consist of the items necessary to uniquely identify the ADM itself. The required metadata items include the following.

| Item | Type | Description | Req. |
|------|------|-------------|------|
|------|------|-------------|------|

| | | | |
|-------------------|-----|--|---|
| Name | STR | The human-readable name of the ADM. | Y |
| Version | STR | Version of the ADM encoded as a string. | Y |
| OID Nickname N | OID | ADMs provide an ordered list of nicknames that can be used by other MIDs in the ADM definition to defined compressed OIDs. There can an arbitratry number of nicknames defined for an ADM. | N |

Table 4: ADM Terminology

[6.3.2.](#) ADM Information Capture

The ADM Data Section consist of all components in the "data" category associated with the managed application or protocol. The information that must be provided for each of these items is as follows.

Name

Every component in an ADM MUST be given a human-readable, consistent name that unquely identifies the component in the context of the application or protocol. These names will be used by human-computer interfaces for manipulating components.

MID

The managed identifier that describes this data item. MIDs in components identified by an ADM MUST NOT contain an issuer or a tag value. In cases where a partial OID is specified, the ADM OID prefix is presumed as the base. In cases where the OID is parameterized, the parameter values are not included in the MID definition in the ADM as parameters are provided at runtime by implementations.

OID

A human-readable version of the OID encapsulated in the MID for the component (e.g., 1.2.3.4). When a nickname is used to represent an compressed OID, the nickname enumeration is included

in this field enclosed by square brackets. For example, if OID nickname 0 refers to the OID prefix 1.2.3.4.5, then the OID 1.2.3.4.5.6 may be listed more compactly as [0].6

Description

Every component in an ADM MUST be given a human-readable, consistent description that provides a potential user with a compact, effective summary of the component.

Type

For components that evaluate to a data value, the data type for that value must be represented.

Parameters

For components with a parameterized OID, the ADM MUST provide the expected number of parameters. A value of 0 indicates that the OID has no parameters and MUST NOT be used for any MID which has a parameterized OID. When omitted, the number of parameters is considered 0.

Parameter N Name

Each parameter of a parameterized component must be given a name.

Parameter N Description

Each parameter of a parameterized component must be given a summary that describes how the parameter will be used by the application or protocol.

Parameter N Type

Each parameter of a parameterized component must be given a type that describes the structure capturing the parameter value. Notably, while parameters in the OID form something similar to a function prototype, there is no sense of function call or stack and therefore all parameters should be considered as pass-by-value.

[7.](#) DTNMP Agent Application Data Model

[7.1.](#) Data Definitions

This section provides the ADM for a DTNMP Agent. This ADM may eventually be removed from the DTNMP specification into its own standards document. All DTNMP agents MUST support the items described in this section.

Internet-Draft

DTNMP

December 2014

[7.2.](#) Metadata Definitions

| Item | Type | Value | Comment |
|-------------------|------|--------------------|---|
| Name | STR | DTNMP Agent ADM | |
| Version | STR | 2014-12-31 | |
| OID Nickname 0 | OID | 1.1 | 1.1 is currently used only as a non-operational example of an ADM base. |

Table 5: DTNMP Agent Metadata

[7.3.](#) Data Definitions

The DTNMP Agent ADM defines the following atomic data definitions that represent the set of data that MUST be collected by any DTNMP agent implementation.

[7.3.1.](#) Atomic Data

Internet-Draft

DTNMP

December 2014

| Name | MID | OID | Description | Type |
|------------------|----------|---------|--------------------------------------|------|
| DefinedReports | 0x800200 | [0].0.0 | # Reports Defined | INT |
| SentReports | 0x800201 | [0].0.1 | # Reports Sent | INT |
| DefinedTimeRules | 0x800202 | [0].0.2 | # Active Time-Based Production Rules | INT |
| RunTimeRules | 0x800203 | [0].0.3 | # Run Time-Based Production Rules | INT |
| DefinedConsts | 0x800204 | [0].0.4 | # Constants Defined | INT |
| DefinedCustom | 0x800205 | [0].0.5 | # Computed Data Definitions | INT |
| DefinedMacros | 0x800206 | [0].0.6 | # Macros Defined | INT |
| RunMacros | 0x800207 | [0].0.7 | # Macros Run | INT |
| DefinedCtrls | 0x800208 | [0].0.8 | # Controls Defined | INT |
| RunCtrls | 0x800209 | [0].0.9 | # Controls Defined | INT |

Table 6: DTNMP Agent Atomic Data

[7.3.2.](#) Computed Data

The DTNMP Agent ADM does not define any computed data items.

[7.3.3.](#) Reports

| Name | MID | OID | Description | Type |
|------------|----------|---------|---------------------------------|------|
| FullReport | 0x880220 | [0].2.0 | Report of all atomic data items | DC |

Table 7: DTNMP Agent Reports

[7.4.](#) Operators

This section describes the standard set of operators available to all DTNMP agents. Applications and protocols do not need to redefine these operators, as they may be used in any expressions evaluated by any agent.

Internet-Draft

DTNMP

December 2014

| Name | MID | OID | Description |
|------|----------|---------|----------------|
| + | 0x830260 | [0].6.0 | Addition |
| - | 0x830261 | [0].6.1 | Subtraction |
| * | 0x830262 | [0].6.2 | Multiplication |
| / | 0x830263 | [0].6.3 | Division |
| % | 0x830264 | [0].6.4 | Modulo |
| ^ | 0x830265 | [0].6.5 | Exponentiation |
| & | 0x830266 | [0].6.6 | Bitwise AND |
| | 0x830267 | [0].6.7 | Bitwise OR |
| # | 0x830268 | [0].6.8 | Bitwise XOR |
| ~ | 0x830269 | [0].6.9 | Bitwise NOT |

| | | | |
|-----|-----------|----------|--------------------------|
| && | 0x83026A | [0].6.A | Logical AND |
| | 0x83026B | [0].6.B | Logical OR |
| ## | 0x83026C | [0].6.C | Logical XOR |
| ! | 0x83026D | [0].6.D | Logical NOT |
| abs | 0x83026E | [0].6.E | Absolute Value |
| < | 0x83026F | [0].6.F | Less than |
| > | 0x8303610 | [0].6.10 | Greater than |
| <= | 0x8303611 | [0].6.11 | Less than or equal to |
| >= | 0x8303612 | [0].6.12 | Greater than or equal to |
| != | 0x8303613 | [0].6.13 | Not equal |
| == | 0x8303614 | [0].6.14 | Equal to |

Table 8: DTNMP Agent Atomic Data

7.5. Controls

This section describes the controls available for use on any DTNMP agent. Since this section essentially provides a functional specification for the DTNMP agent, it is presented in two sections. First, a summary section provides a quick reference for the controls available on a DTNMP agent. Second, a full control specification provides detailed information on each individual DTNMP agent control.

7.5.1. Control Summary

| Name | MID | OID | Description |
|----------|----------|---------|------------------------|
| ListADMs | 0x810230 | [0].3.0 | List all ADMs known to |

| | | | | |
|----------------|----------|---------|--|---|
| | | | | the agent. |
| ListAtomicIDs | 0x810231 | [0].3.1 | | List all Atomic MIDs known to the agent. |
| DescAtomicData | 0xC10232 | [0].3.2 | | Dump information for given Atomic MIDs. |
| AddCompData | 0xC10233 | [0].3.3 | | Define a computed data definition on the agent. |
| DelCompData | 0xC10234 | [0].3.4 | | Remove a computed data definition from the agent. |
| ListCompData | 0x810235 | [0].3.5 | | List known computed data MIDs. |
| DescCompData | 0xC10236 | [0].3.6 | | Dump information for given computed data MIDs. |
| AddRptDef | 0xC10237 | [0].3.7 | | Define a custom report. |
| DelRptDef | 0xC10238 | [0].3.8 | | Forget a custom report. |
| ListRpts | 0x810239 | [0].3.9 | | List known custom report MIDs. |
| DescRpts | 0xC1023A | [0].3.A | | Dump information for given custom report MIDs. |
| ListOps | 0x81023B | [0].3.B | | List known operator IDs. |

| | | | |
|-----------|----------|---------|---|
| DescOps | 0xC1023C | [0].3.C | Dump information for given operator MIDs. |
| ListCtrls | 0x81023D | [0].3.D | List known custom control MIDs. |
| DescCtrls | 0xC1023E | [0].3.E | Dump information for given controls MIDs. |

| | | | |
|----------------|-----------|----------|---|
| AddMacroDef | 0xC1023F | [0].3.F | Define a custom macro. |
| DelMacroDef | 0xC103310 | [0].3.10 | Forget a custom macro. |
| ListMacros | 0x8103311 | [0].3.11 | List known custom macro MIDs. |
| DescMacros | 0xC103312 | [0].3.12 | Dump information for given custom macro MIDs. |
| AddTimeRule | 0xC103313 | [0].3.13 | Define a time-based prod rule. |
| DelTimeRule | 0xC103314 | [0].3.14 | Forget a time-based prod rule. |
| ListTimeRules | 0x8103315 | [0].3.15 | List known time-based rules. |
| DescTimeRules | 0xC103316 | [0].3.16 | Dump information for given time-based rules. |
| AddStateRule | 0xC103317 | [0].3.17 | Define a state-based prod rule. |
| DelStateRule | 0xC103318 | [0].3.18 | Forget a state-based prod rule. |
| ListStateRules | 0x8103319 | [0].3.19 | List known state-based rules. |
| DescStateRules | 0xC10331A | [0].3.1A | Dump information for given state-based rules. |

Table 9: DTNMP Agent Controls Summary

[7.5.2.1.](#) ListADMs

This control causes the agent to produce a report detailing the list of all ADMs configured on the agent and available for use.

This control takes no parameters.

This control causes a report to be generated with the following format.

ListADMs Report Format

| | | | |
|---------------|------------|---------|------------|
| +-----+-----+ | | +-----+ | |
| # ADMs | ADM 1 Name | ... | ADM N Name |
| [SDNV] | [STR] | | [STR] |
| +-----+-----+ | | +-----+ | |

Figure 18

ADMs

The number of ADMs known to the agent.

ADM [x] Name

For each ADM, it's human readable name.

[7.5.2.2.](#) ListAtomicIDs

This control causes the agent to produce a list containing the MID of every atomic data item understood by the agent.

This control takes no parameters.

This control causes a report to be generated with the following format.

ListAtomicIDs Report Format

| | |
|------------|--|
| +-----+ | |
| Atomic IDs | |
| [MC] | |
| +-----+ | |

Figure 19

Atomic IDs

A MID Collection of the MIDs of each atomic data definition known to the agent.

[7.5.2.3.](#) DescAtomicData

This control causes the agent to produce a detailed description of the requested atomic data definitions.

This control takes 1 parameter in the following format.

DescAtomicData Parameters

```
+-----+
| Atomic IDs |
|   [MC]   |
+-----+
```

Figure 20

Atomic IDs

The list of MIDs identifying the atomic IDs to be described.

This control causes a report to be generated with the following format.

DescAtomicData Report Format

```
+-----+-----+-----+-----+-----+
| # IDs | ID1 Name | ID1 Type | ... | IDn Name | IDn Type |
| [SDNV] | [STR]   | [SDNV]   |     | [STR]   | [SDNV]   |
+-----+-----+-----+-----+-----+
```

Figure 21

IDs

The number of atomic data descriptions returned.

ID[n] Name

The name of the nth returned atomic data item.

ID[n] Type

Type enumeration for the nth atomic data item.

[7.5.2.4.](#) AddCompData

This control configures a new computed data definition on the agent.

A computed data item uses an expression to assign a data value.

Internet-Draft

DTNMP

December 2014

This control takes 3 parameters in the following format.

AddCompData Parameters

| | | | |
|---------------------|--------|--------|--|
| +-----+-----+-----+ | | | |
| Def ID | Def | Type | |
| [MID] | [EXPR] | [SDNV] | |
| +-----+-----+-----+ | | | |

Figure 22

Def ID

The MID value identifying the new computed data definition.

Def

The expression used to calculate the value for the new computed data item.

Type

The data type for the resultant computed data value.

This control causes no report to be produced.

[7.5.2.5](#). DelCompData

This control removes a computed data definition from the agent.

This control takes 1 parameter in the following format.

DelCompData Parameters

| | |
|---------------|--|
| +-----+ | |
| IDs To Remove | |
| [MC] | |
| +-----+ | |

Figure 23

IDs To Remove

The list of computed data items to be removed from the agent.

This control causes no report to be produced.

[7.5.2.6.](#) ListCompData

This control causes the agent to produce a list containing the MID of every computed data definition understood by the agent.

This control takes no parameters.

This control causes a report to be generated with the following format.

ListCompData Report Format

```
+-----+
| Custom IDs |
|   [MC]   |
+-----+
```

Figure 24

Custom IDs

The collection of identifiers for those computed data definitions known to the agent.

[7.5.2.7.](#) DescCompData

This control causes the agent to produce a detailed description of the requested computed data definitions.

This control takes 1 parameter in the following format.

DescCompData Parameters

```
+-----+
| Comp IDs |
|   [MC]   |
+-----+
```

Figure 25

Comp IDs

The MID collection identifying the computed data items to be described.

This control causes a report to be generated with the following format.

DescCompData Report Format

| | | | | | | | |
|---------|---------|---------|---------|-----|---------|---------|---------|
| +-----+ | +-----+ | +-----+ | +-----+ | | +-----+ | +-----+ | +-----+ |
| # IDs | C1 ID | C1 Def | C1 Type | ... | Cn ID | Cn Def | Cn Type |
| [SDNV] | [MID] | [EXPR] | [SDNV] | | [MID] | [EXPR] | [SDNV] |
| +-----+ | +-----+ | +-----+ | +-----+ | | +-----+ | +-----+ | +-----+ |

Figure 26

IDs

The number of computed data definitions in the report.

C[n] ID

The MID of the nth computed data definition.

C[n] Def

The expression used to calculate the value of the Nth computed data item.

C[n] Type

The data type of the value of the Nth computed data item.

[7.5.2.8](#). AddRptDef

This control configures a new custom report definition on the agent.

A custom report assigns a single MID value to represent an ordered collection of other MID values, with some administrative information that identifies what other nodes in the network may request and process this report.

This control takes 2 parameters in the following format.

AddRptDef Parameters

| | |
|---------------|------------|
| +-----+-----+ | |
| RPT ID | Definition |
| [MID] | [MC] |
| +-----+-----+ | |

Figure 27

Rpt ID

The MID for the new report.

Definition

The ordered set of MIDs representing the contents of the report.

This control causes no report to be produced.

[7.5.2.9](#). DelRptDef

This control removes a custom report definition from the agent.

This control takes 1 parameters in the following format.

DelRptDef Parameters

| | |
|---------------|--|
| +-----+-----+ | |
| IDs To Remove | |
| [MC] | |
| +-----+-----+ | |

Figure 28

IDs to Remove

The collection of MIDs identifying the report definitions to remove from the agent.

This control causes no report to be produced.

[7.5.2.10.](#) ListRpts

This control causes the agent to produce a list containing the MID of every report definition understood by the agent.

This control takes no parameters.

This control causes a report to be generated with the following format.

ListRpts Report Format

```
+-----+  
| Reports |  
|  [MC]  |  
+-----+
```

Figure 29

Reports

The collection of MIDs representing custom report IDs known to the agent.

[7.5.2.11.](#) DescRpts

This control causes the agent to produce a detailed description of the requested report definitions.

This control takes 1 parameter in the following format.

DescRpts Parameters

```
+-----+  
| Report IDs |
```



```

|      [MC]      |
+-----+

```

Figure 30

Report IDs

The collection of MIDs identifying the report definitions to be described.

This control causes a report to be generated with the following format.

DescRpts Report Format

```

+-----+-----+-----+          +-----+-----+
| # IDs | RPT 1 ID | RPT 1 Def | ... | RPT N ID | RPT N Def |
| [SDNV] | [MID]   | [MC]     |     | [MID]   | [MC]     |
+-----+-----+-----+          +-----+-----+

```

Figure 31

IDs

The number of report definitions in this report.

RPT[n] ID

The MID of the nth report definition.

RPT[n] Def

The ordered collection of IDs comprising the Nth report definition.

[7.5.2.12](#). ListOps

This control causes the agent to produce a list containing the MID of every operator understood by the agent.

This control takes no parameters.

This control causes a report to be generated with the following format.

ListOps Report Format

| | |
|---------|--|
| +-----+ | |
| OPS | |
| [MC] | |
| +-----+ | |

Figure 32

OPS

The collection of MIDs for every op known by the agent.

[7.5.2.13.](#) DescOps

This control causes the agent to produce a detailed listing of the requested operator definitions.

This control takes 1 parameter in the following format.

DescOps Parameters

| | |
|---------|--|
| +-----+ | |
| OP IDs | |
| [MC] | |
| +-----+ | |

Figure 33

OP IDs

The set of operators to be described.

This control causes a report to be generated with the following format.

DescOps Report Format

| | | | | | | | |
|---------|---------|-----------|-----|---------|-----------|--|--|
| +-----+ | | | | +-----+ | | | |
| # OPs | OP 1 ID | OP 1 Desc | ... | OP N ID | OP N Desc | | |
| [SDNV] | [MID] | [STR] | | [MID] | [STR] | | |
| +-----+ | | | | +-----+ | | | |

Figure 34

OPs

The number of operator definitions in the report.

OP[n] ID
 The MID of the nth operator definition.

OP[n] Def
 The description of the nth operator.

[7.5.2.14.](#) ListCtrls

This control causes the agent to produce a list containing the MID of every control understood by the agent.

This control takes no parameters.

This control causes a report to be generated with the following format.

ListCtrls Report Format

```
+-----+
| CTRLs |
| [MC]  |
+-----+
```

Figure 35

CTRLs
 The collection of MIDs of controls known to the agent.

[7.5.2.15.](#) DescCtrls

This control causes the agent to produce a detailed listing of the requested control definitions.

This control takes 1 parameter in the following format.

DescCtrls Parameters

```
+-----+
| CTRL IDs |
|   [MC]   |
+-----+
```

Figure 36

CTRL IDs
 The set of controls to be described.

Internet-Draft

DTNMP

December 2014

This control causes a report to be generated with the following format.

DescCtrls Report Format

| | | | | | |
|---------|----------|------------|-----|----------|------------|
| +-----+ | +-----+ | +-----+ | | +-----+ | +-----+ |
| # CTRLs | CTRL1 ID | CTRL1 Desc | ... | CTRLN ID | CTRLN Desc |
| [SDNV] | [MID] | [STR] | | [MID] | [STR] |
| +-----+ | +-----+ | +-----+ | | +-----+ | +-----+ |

Figure 37

CTRLs

The number of control definitions in the report.

CTRL[n] ID

The MID of the nth control definition.

CTRL[n] Desc

The description of the Nth control definition, to include information on parameters.

[7.5.2.16](#). AddMacroDef

This control configures a new custom macro definition on the agent. A macro is a series of controls that should be run in sequence.

This control takes 3 parameters in the following format.

AddMacroDef Parameters

| | | |
|------------|----------|---------|
| +-----+ | +-----+ | +-----+ |
| Macro Name | Macro ID | Def |
| [STR] | [MID] | [MC] |
| +-----+ | +-----+ | +-----+ |

Figure 38

Macro Name

The descriptive name for the macro.

Macro ID

The MID value identifying the new macro.

Def

The ordered set of controls/macros that are run sequentially during macro execution.

This control causes no report to be produced.

[7.5.2.17](#). DelMacroDef

This control removes a custom macro definition from the agent.

This control takes 1 parameters in the following format.

DelMacroDef Parameters

```
+-----+
| IDs To Remove |
|      [MC]      |
+-----+
```

Figure 39

IDs To Remove

The collection of MIDs identifying the macros to be removed from the agent.

This control causes no report to be produced.

[7.5.2.18](#). ListMacros

This control causes the agent to produce a list containing the MID of every custom macro definition understood by the agent.

This control takes no parameters.

This control causes a report to be generated with the following format.

ListMacros Report Format

```
+-----+
| Macro IDs |
|   [MC]   |
+-----+
```

Figure 40

Macro IDs

The list of macro definitions known to the agent.

[7.5.2.19](#). DescMacros

This control causes the agent to produce a detailed listing of the requested macro definitions.

This control takes 1 parameter in the following format.

DescMacros Parameters

```
+-----+
| Macro IDs |
|   [MC]   |
+-----+
```

Figure 41

Macro IDs

The collection of MIDs identifying the macros to be described.

This control causes a report to be generated with the following format.

DescMacros Report Format

```
+-----+-----+-----+-----+      +-----+-----+-----+
```

| # Macros | M1 Name | M1 ID | M1 Def | ... | Mn Name | Mn ID | Mn Def |
|----------|---------|-------|--------|-----|---------|-------|--------|
| [SDNV] | [STR] | [MID] | [MC] | | [STR] | [MID] | [MC] |
| + | + | + | + | + | + | + | + |

Figure 42

Macros

The number of macro definitions in the report.

M[n] Name

The name of the nth macro definition.

M[n] ID

The MID of the nth macro definition. v

M[n] Def

The expression used to calculate the value of the Nth computed data item.

[7.5.2.20](#). AddTimeRule

This control configures a new time-based production rule on the agent. A time-based production rule instructs the agent to produce a report comprising a fixed set of component values periodically over time. The component values may represent any type of data value, including atomic data, computed data, or reports.

This control takes 4 parameters in the following format.

AddTimeRule Parameters

| Start | Period (s) | Count | Results |
|-------|------------|--------|---------|
| [TS] | [SDNV] | [SDNV] | [MC] |
| + | + | + | + |

Figure 43

Start
The time at which the production should commence.

Period
The number of seconds to wait between report message generation.

Count
The number of reports to be generated by this configuration. The special value of 0 indicates production should continue indefinitely.

Results
The collection of MIDs to be included in the report.

No report is produced in response to this individual command. The configured reports will be produced in accordance to the configured time period.

[7.5.2.21](#). DelTimeRule

This control removes a time-based production rule from the agent.

This control takes 1 parameters in the following format.

DelTimeRule Parameters

```
+-----+
| Rule IDs |
|   [MC]   |
+-----+
```

Figure 44

Rule IDs
The collection of MIDs identifying the rules to be removed from the agent.

This control causes no report to be produced.

[7.5.2.22.](#) ListTimeRules

This control causes the agent to produce a list containing the MID of every time-based production rule understood by the agent.

This control takes no parameters.

This control causes a report to be generated with the following format.

ListTimeRules Report Format

```
+-----+
| Rule IDs |
|   [MC]   |
+-----+
```

Figure 45

Rule IDs

The collection of MIDs identifying the time-based rules known to the agent.

[7.5.2.23.](#) DescTimeRules

This control causes the agent to produce a detailed listing of the requested time-based production rules.

This control takes 1 parameter in the following format.

DescTimeRules Parameters

```
+-----+
| Rule IDs |
|   [MC]   |
+-----+
```

+-----+

Figure 46

Rule IDs

The collection of MIDs identifying the time-based rules to be described.

This control causes a report to be generated with the following format.

DescTimeRules Report Format

| | | | | | | |
|---------------------------------------|-------|----------|-----------|----------|-----------|-----|
| +-----+-----+-----+-----+-----+-----+ | | | | | | |
| # Rules | R1 ID | R1 START | R1 PERIOD | R1 Count | R1 Result | ... |
| [SDNV] | [MID] | [TS] | [SDNV] | [SDNV] | [MC] | |
| +-----+-----+-----+-----+-----+-----+ | | | | | | |

Figure 47

Rules

The number of time-based rule definitions in the report.

R[n] ID

The MID identifier for the nth time-based rule definition.

R[n] Start

The time at which the nth time-based rule production should commence.

R[n] Period

The number of seconds to wait between running the nth time-based rule.

R[n] Count

The number of reports to be generated by the nth time-based report. The special value of 0 indicates production should continue indefinitely.

R[n] Results

The collection of MIDs to be included in the report generated by the nth time-based rule.

[7.5.2.24.](#) AddStateRule

This control configures a new state-based production rule on the agent. A state-based production rule instructs the agent to produce a report comprising a fixed set of component values periodically whenever the expression evaluates to true. The component values may represent any type of data value, including atomic data, computed data, or reports.

This control takes 4 parameters in the following format.

AddStateRule Parameters

| | | | |
|-------|--------|--------|---------|
| Start | State | Count | Results |
| [TS] | [PRED] | [SDNV] | [MC] |

Figure 48

- Start**
The time at which the production should commence.
- State**
The expression which, when non-zero, causes the report to be generated.
- Count**
The number of reports to be generated by this configuration. The special value of 0 indicates production should continue indefinitely.
- Results**
The collection of MIDs to be included in the report.
- No report is produced in response to this individual command.

[7.5.2.25.](#) DelStateRule

This control removes a state-based production rule from the agent.

This control takes 1 parameters in the following format.

Internet-Draft

DTNMP

December 2014

DelStateRule Parameters

```

+-----+
| Rule IDs |
|   [MC]   |
+-----+

```

Figure 49

Rule IDs

The collection of MIDs identifying the rules to be removed from the agent.

This control causes no report to be produced.

[7.5.2.26](#). ListStateRules

This control causes the agent to produce a list containing the MID of every state-based production rule understood by the agent.

This control takes no parameters.

This control causes a report to be generated with the following format.

ListStateRules Report Format

```

+-----+
| Rule IDs |
|   [MC]   |
+-----+

```

Figure 50

Rule IDs

The collection of MIDs identifying the state-based rules known to the agent.

[7.5.2.27](#). DescStateRules

This control causes the agent to produce a detailed listing of the requested state-based production rules.

This control takes 1 parameter in the following format.

DescStateRules Parameters

```
+-----+
| Rule IDs |
|  [MC]   |
+-----+
```

Figure 51

Rule IDs

The collection of MIDs identifying the state-based rules to be described.

This control causes a report to be generated with the following format.

DescStateRules Report Format

```
+-----+-----+-----+-----+-----+-----+
| # Rules | R1 ID | R1 START | R1 PERIOD | R1 Count | R1 Result | ...
| [SDNV] | [MID] | [TS]     | [SDNV]   | [SDNV]   | [MC]     |
+-----+-----+-----+-----+-----+-----+
```

Figure 52

Rules

The number of state-based rule definitions in the report.

R[n] ID

The MID identifier for the nth state-based rule definition.

R[n] Start

The time at which the nth state-based rule production should commence.

R[n] Period

The number of seconds to wait between running the nth state-based rule.

R[n] Count

The number of reports to be generated by the nth state-based report. The special value of 0 indicates production should continue indefinitely.

R[n] Results

The collection of MIDs to be included in the report generated by the nth state-based rule.

[8.](#) IANA Considerations

At this time, this protocol has no fields registered by IANA.

[9.](#) Security Considerations

Transport security is handled by the transport layer, for example the Bundle Security Protocol [[RFC6257](#)] when using the Bundle Protocol [[RFC5050](#)].

Finer grain application security is done via ACLs which are defined via configuration messages and implementation specific.

[10.](#) Normative References

- [DTNM] Birrane, E. and H. Kruse, "DTN Network management: The Definition and Exchange of Infrastructure Information in High Delay Environments", .
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", [RFC 4838](#), April 2007.
- [RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", [RFC 5050](#), November 2007.

[RFC6256] Eddy, W. and E. Davies, "Using Self-Delimiting Numeric Values in Protocols", [RFC 6256](#), May 2011.

[RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", [RFC 6257](#), May 2011.

[tolerance] Birrane, E., Burleigh, S., and V. Cerf, "Defining Tolerance: Impacts of Delay and Disruption when Managing Challenged Networks", 2001.

Authors' Addresses

Edward J. Birrane
Johns Hopkins Applied Physics Laboratory

Email: Edward.Birrane@jhuapl.edu

Birrane & Ramachandran Expires July 4, 2015

[Page 55]

Internet-Draft

DTNMP

December 2014

Vignesh Ramachandran
Johns Hopkins Applied Physics Laboratory

Email: Vinny.Ramachandran@jhuapl.edu

