Asynchronous Mgmt Architecture (AMA) & Asynchronous Mgmt Protocol (AMP) Updates

Edward Birrane

Edward.Birrane@jhuapl.edu

443-778-7423

APL
JOHNS HOPKINS UNIVERSITY
Applied Physics Laboratory

AMA: Overview

From draft-birrane-dtn-ama-03

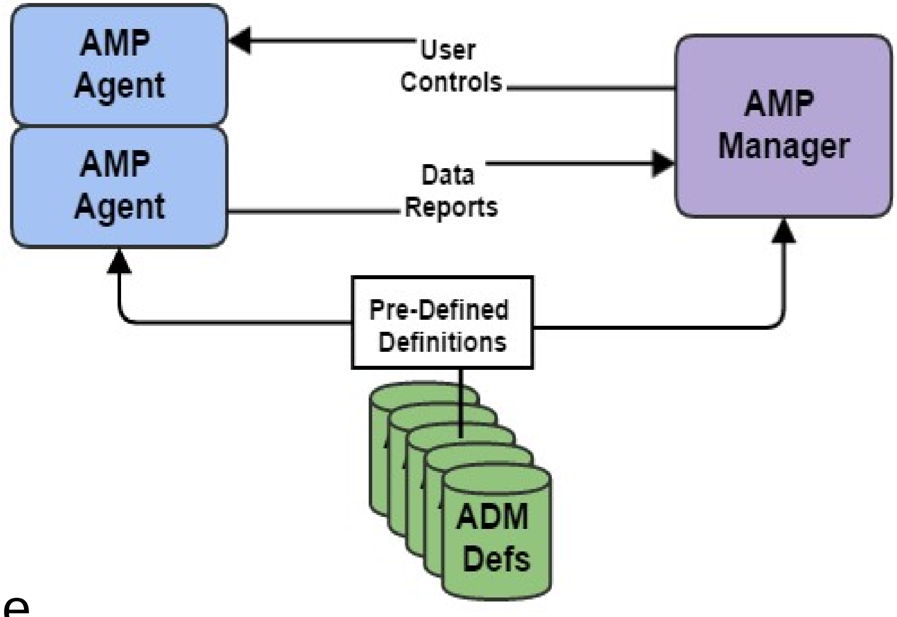
- Service Definitions
  - **Configuration**: Change settings on an Agent.
  - **Reporting**: Receive performance information from an Agent.
  - **Autonomous Parameterized Control**: Change Agent Behavior.
  - **Administration**: Fine-grained access to abilities.

- Desirable Properties
  - **Intelligent Information Push**: Can’t rely on others.
  - **Minimize Message Size**: Increase probability of delivery.
  - **Absolute Data Identification**: pre-shared, global naming when possible.
  - **Custom Data Definition**: Only send minimal necessary data sets.
  - **Autonomous Operation**: Decisions local to Agent based on its config.

AMA: System Model

From draft-birrane-dtn-ama-03

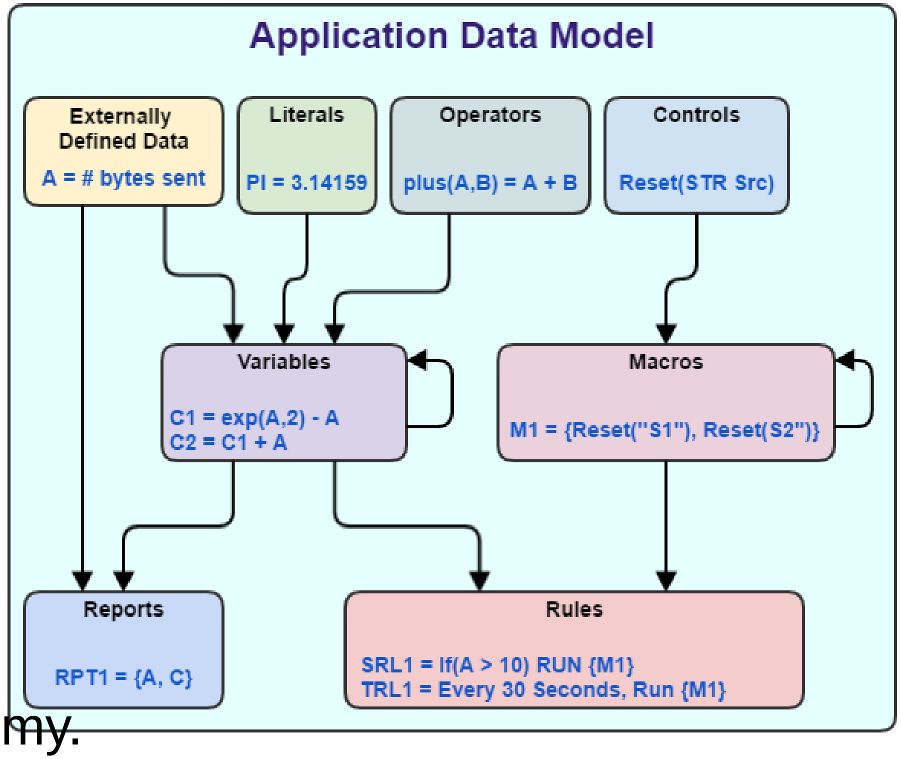
- Agents
  - Run on Managed Devices
  - Configure/Report on devices
  - Heavy autonomy and parameterized control
- Manager(s)
  - Collect/Fuse data from Agents
  - Configure Agent behavior
  - Open-loop control
- ADMs
  - Well-named Data and Controls
  - Superset of MIB
  - Move to describe them in YANG
  - Preconfiguration reduces msg size



AMA: App. Data Model (ADM)

- “Atomic” Elements
  - Solely defined by their ADM.
  - **EDDs**: collected by agents.
  - **Literals:** useful constants.
  - **Ops:** opcodes for math functions.
  - **Ctrls:** opcodes for agent behavior.
- “Variable” Elements
  - Defined by ADM or by User
  - ADM definitions are immutable.
  - **Vars**: strong-typed variables, including a type for “expression”.
  - **Macro**: Ordered set of Ctrls.
  - **Rpts**: Ordered sets of data
  - **Rules:** Time or State based autonomy.

An ADM defined 8 types of data for each application/protocol managed in the AMA.



Application Data Model

Externally Defined Data
A = # bytes sent

Literals
PI = 3.14159

Operators
plus(A,B) = A + B

Controls
Reset(STR Src)

Variables
C1 = exp(A,2) - A
C2 = C1 + A

Macros
M1 = {Reset("S1"), Reset(S2")}

Reports
RPT1 = {A, C}

Rules
SRL1 = If(A > 10) RUN {M1}
TRL1 = Every 30 Seconds, Run {M1}

AMA: Updates

From -02 to -03

- Minor Terminology and Definitions Updates
  - Mostly wordsmithing based on feedback.
  - Clarify distinction between a Report Template, a Report Entry, and a Report as a collection of entries.
  - Some expanded text around parameterization and motivation for the approach.

- No significant issues or limitations with the architecture.
  - No “structural” changes to the architecture.

- Primary focus has been on AMP and associated ADMs.

AMA: TODO

Propose AMA provides the architecture and required functions of a DTN Network Management Protocol.

- Some discussion of terminology
  - Atomic Data vs. Primitive Data vs. Externally Defined Data
  - Computed Data vs Variables
  - Specs out of sync on terminology.
    - *Need a refresh across related drafts to synchronize them.*
- OPs AD feedback
  - Reviewing RESTful NETCONF and YANG Push models
  - Review to date does not seem to change the AMA.
    - *May also not change the AMP, which is considered separately.*
- Request AMA be considered by the WG when it is time to address Network Management for DTNs.

AMP: Overview

AMP being evaluated by space and non-space users. NASA providing an open-source reference implementation in ION.

- Protocol conformant to the architecture/requirements of AMA.
  - Implements Agents, Managers, ADM structures.
  - Defines specific data models to implement AMA structures
  - Defines messages to capture AMA controls/reports/administration
  - Defines on-the-wire encodings
- Data Models
  - Basic Types: Numeric types, strings, etc…
  - Compound Types: BLOBs, (Typed) Data Collections, Tables, Identifiers, Collections, Expressions, Predicates
- Functional Specification
  - AMP Message Groups: Common headers and trailers
  - Three messages: RegisterAgent, PerformControl, DataReport

AMP: Updates (1/2)

From -02 to -03

- Minor Terminology and Definitions Updates
  - Wordsmithing based on feedback.
  - Reduced redundancy between AMP and AMA specs.

- Clarifications
  - Clarified Report Templates vs Report Entries vs Reports.
  - Clarified State vs Time-based Rules.
  - Corrected AMP Epoch time.
  - Added rationale for design of TDCs.
  - Clarified that OID Nicknames are registered values.
  - Clarified OID Parameterization Approach
  - Clarified definition of Variables and their initializing expression.

AMP: Updates (2/2)

From -02 to -03

- Additions/Updates
  - Added Table AMP structure.
  - Added Result Type to Expression structure.
  - Added required levels of Macro nesting.
  - Updated type enumerations.
  - Added allowed numerical promotions
  - Added rules for numeric conversions
  - Updated format of DataReport message.
- Removals
  - Removed draft design of N of M counts for SRLs.
  - Removed enable/disable from SRL and TRL structures

AMP: TODO

From -02 to -03

- Upcoming Spec Changes
  - How best to add N of M and enabled/disabled to SRL/TRLs
  - Change TDC column IDs to be of any type, not just string.
  - Add Access Control Lists (ACLs) and describe behavior.
  - Transition to CBOR for encoding.
  - Add guidance in ADM section on when to define TABLEs versus EDDs vs Controls that return data.
  - Should AMP specify a wire encoding?

- More Review from Reference Implementations
  - Continued support of reference implementation efforts
  - At last count there were 4 separate implementation efforts
    - *Discussions on 2 additional efforts.*

AMA/AMP Related Specifications

- Core Specs
  - AMA: draft-birrane-dtn-ama-03
  - AMP: draft-birrane-dtn-amp-03
- ADMs
  - AMP Agent ADM: draft-birrane-dtn-adm-agent-02
  - BPSEC ADM: draft-birrane-dtn-adm-bpsec-00
  - BP ADM: draft-birrane-dtn-adm-bp-00
  - YANG profile for ADMs: draft-bsipos-dtn-amp-yang-01
- Other:
  - AMP Manager SQL Schema: draft-birrane-dtn-ampmgr-sql-00

Current Status

NASA building out AMP for deployment to ISS and other infusion targets

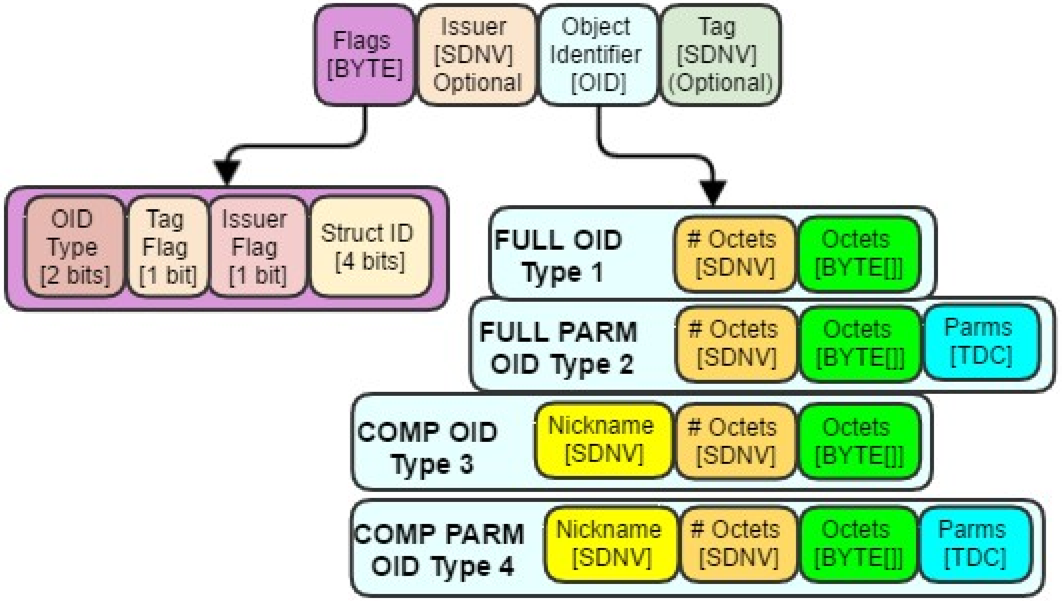
- Reference implementation in ION open source this year.
  - Supporting AMP protocol messages, Agent, BP, BPSEC ADMss.
- NASA supporting AMA/AMP ongoing work
  - Writing ADMs for BP, BSP, CGR, LTP, and ION.
- Several non-NASA efforts ongoing.
  - AMP is not directly tied to BP or DTN, though it is very helpful for DTN use cases.
- Finalizing AMA and AMP specs for consideration in DTNWG
  - As novel intersection between performance monitoring and safing autonomy
  - Meeting with OPS AD people as they are identified to discuss AMP vs RESTful NETCONF and YANG Push.

Backup

AMP: Key Concept: MIDs

Every AMP structure identified and parameterized by a Managed Identifier (MID).

- Concept
  - Every AMP structure is identified by a MID.
  - Simplifies processing.
  - Hardware acceleration.
- Issuer/Tag
  - MIDs defined in ADM. have no Issuer/Tag.
  - User-defined MIDs must have issuer ID.
  - Tags always optional.
- Parameters
  - Captured in the MID itself.



Flags [BYTE]
Issuer [SDNV] Optional
Object Identifier [OID]
Tag [SDNV] (Optional)

OID Type [2 bits]
Tag Flag [1 bit]
Issuer Flag [1 bit]
Struct ID [4 bits]

FULL OID Type 1
Octets [SDNV]
Octets [BYTE[]]

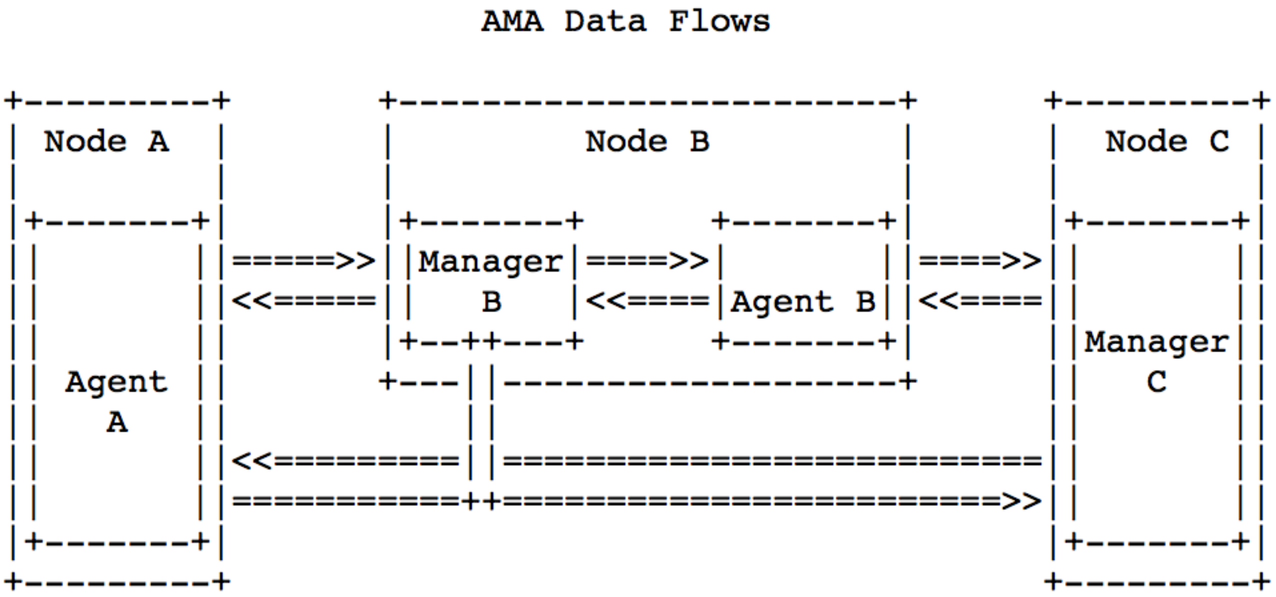
FULL PARM OID Type 2
Octets [SDNV]
Octets [BYTE[]]
Parms [TDC]

COMP OID Type 3
Nickname [SDNV]
Octets [SDNV]
Octets [BYTE[]]

COMP PARM OID Type 4
Nickname [SDNV]
Octets [SDNV]
Octets [BYTE[]]
Parms [TDC]

Control MID: Generate Agent ADM Report
0xc304010903021517050182030100020100
Control MID: List VARs known to Agent
0x83040103
Var. MID: User-Defined UINT variable.
0x110103010203

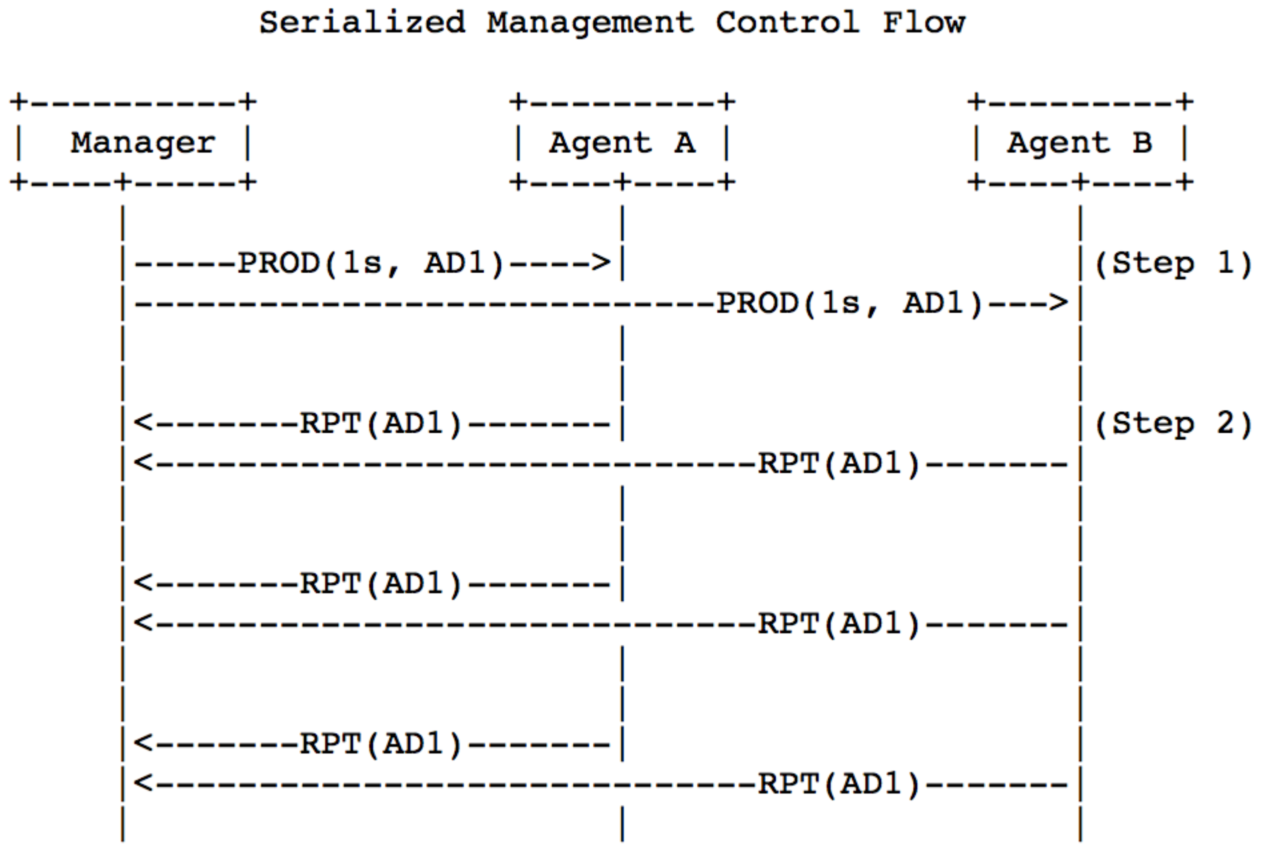
AMA Roles and Responsibilities



```
                       AMA Data Flows

+---------+       +------------------------+      +---------+
| Node A  |       |         Node B         |      |  Node C |
|         |       |                        |      |         |
|+-------+|       |+-------+      +-------+|      |+-------+|
||       ||=====>>||Manager|====>>|       ||====>>||       ||
||       ||<<=====||   B   |<<====|Agent B||<<====||       ||
||       ||       |+--++---+      +-------+|      ||Manager||
|| Agent ||       +---||-------------------+      ||   C   ||
||   A   ||           ||                          ||       ||
||       ||<<=========||==========================||       ||
||       ||===========++========================>>||       ||
|+-------+|                                       |+-------+|
+---------+                                       +---------+
```

AMA Basic Data Flow

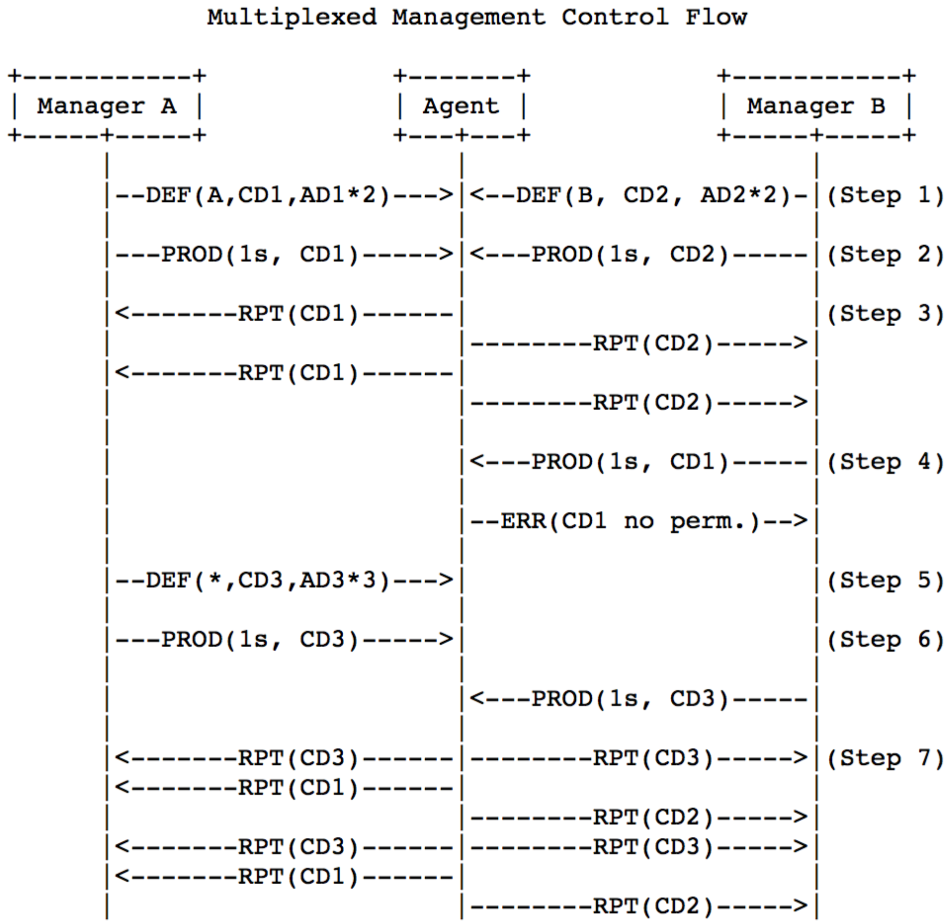


```
              Serialized Management Control Flow

  +----------+            +---------+           +---------+
  |  Manager |            | Agent A |           | Agent B |
  +----+-----+            +----+----+           +----+----+
       |                       |                     |
       |-----PROD(1s, AD1)---->|                     |(Step 1)
       |----------------------------PROD(1s, AD1)--->|
       |                       |                     |
       |                       |                     |
       |<-------RPT(AD1)-------|                     |(Step 2)
       |<-----------------------------RPT(AD1)-------|
       |                       |                     |
       |                       |                     |
       |<-------RPT(AD1)-------|                     |
       |<-----------------------------RPT(AD1)-------|
       |                       |                     |
       |                       |                     |
       |<-------RPT(AD1)-------|                     |
       |<-----------------------------RPT(AD1)-------|
       |                       |                     |
```

In a simple network, a Manager interacts with multiple Agents.

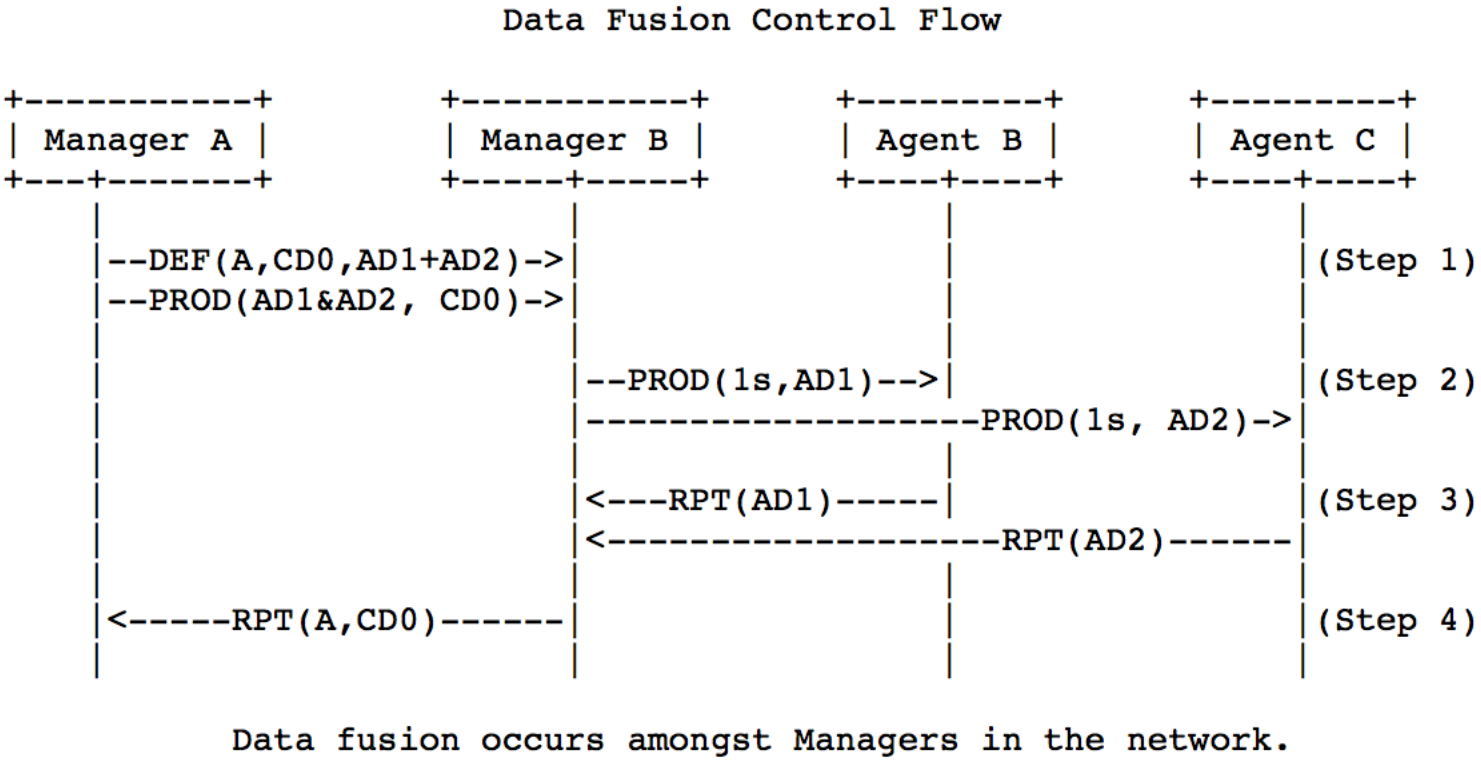
AMA Multi-Manager Flow



```
             Multiplexed Management Control Flow

+-----------+               +-------+            +-----------+
| Manager A |               | Agent |            | Manager B |
+-----+-----+               +---+---+            +-----+-----+
      |                         |                      |
      |--DEF(A,CD1,AD1*2)--->|<--DEF(B, CD2, AD2*2)-|(Step 1)
      |                      |                      |
      |---PROD(1s, CD1)----->|<---PROD(1s, CD2)-----|(Step 2)
      |                      |                      |
      |<-------RPT(CD1)------|                      |(Step 3)
      |                      |--------RPT(CD2)----->|
      |<-------RPT(CD1)------|                      |
      |                      |--------RPT(CD2)----->|
      |                      |                      |
      |                      |<---PROD(1s, CD1)-----|(Step 4)
      |                      |                      |
      |                      |--ERR(CD1 no perm.)-->|
      |                      |                      |
      |--DEF(*,CD3,AD3*3)--->|                      |(Step 5)
      |                      |                      |
      |---PROD(1s, CD3)----->|                      |(Step 6)
      |                      |                      |
      |                      |<---PROD(1s, CD3)-----|
      |                      |                      |
      |<-------RPT(CD3)------|--------RPT(CD3)----->|(Step 7)
      |<-------RPT(CD1)------|                      |
      |                      |--------RPT(CD2)----->|
      |<-------RPT(CD3)------|--------RPT(CD3)----->|
      |<-------RPT(CD1)------|                      |
      |                      |--------RPT(CD2)----->|
```

AMA Data Fusion Flow



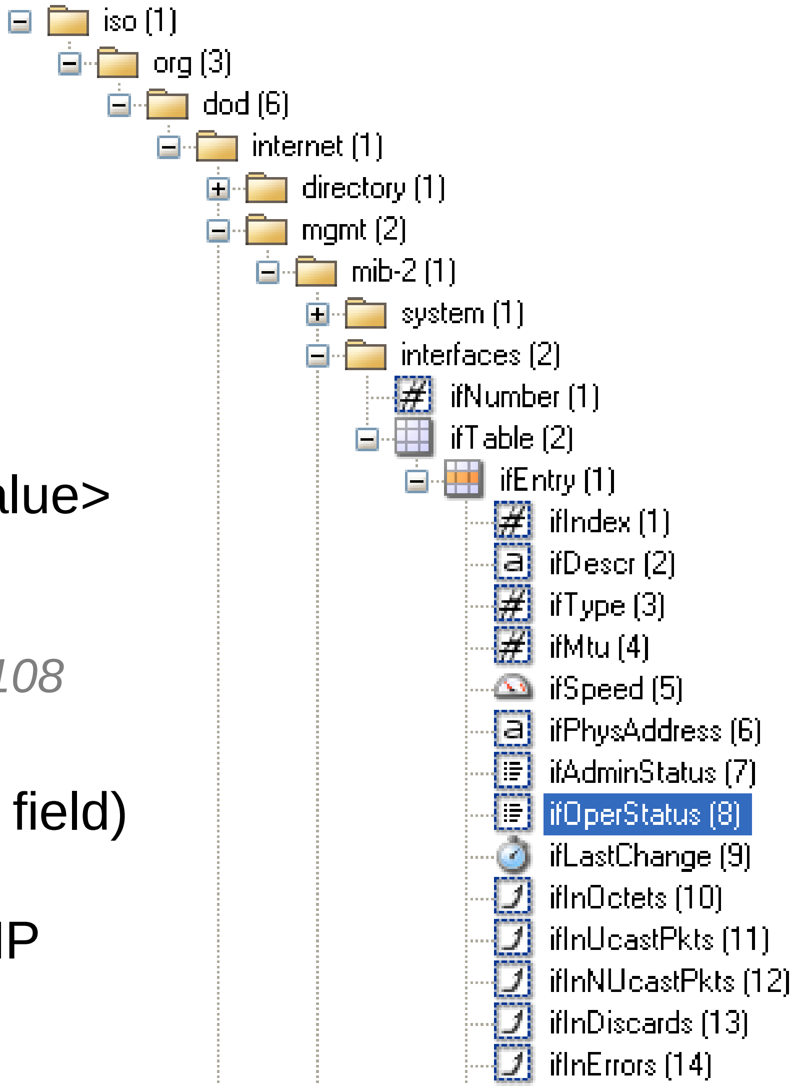
```
                       Data Fusion Control Flow

+-----------+          +-----------+      +---------+      +---------+
| Manager A |          | Manager B |      | Agent B |      | Agent C |
+---+-------+          +-----+-----+      +----+----+      +----+----+
    |                      |                 |                |
    |--DEF(A,CD0,AD1+AD2)->|                 |                |(Step 1)
    |--PROD(AD1&AD2, CD0)->|                 |                |
    |                      |                 |                |
    |                      |--PROD(1s,AD1)-->|                |(Step 2)
    |                      |-------------------PROD(1s, AD2)->|
    |                      |                 |                |
    |                      |<---RPT(AD1)-----|                |(Step 3)
    |                      |<-------------------RPT(AD2)------|
    |                      |                 |                |
    |<-----RPT(A,CD0)------|                 |                |(Step 4)
    |                      |                 |                |

             Data fusion occurs amongst Managers in the network.
```

Compatibility with existing mechanism

- SNMP Uses OIDs as IDs
  - Global, Managed Tree Structure
    - *“Path to data” is concatenation of #s.*
    - *ifSpeed = 1.3.6.1.2.1.2.2.1.8*
  - Supports Binary Encoding (BER)
    - *Compress first 2 #s: 1.3 => 43*
    - *SDNV-encode rest*
  - SNMP Identifier: <type> <length> <value>
    - *Type 6 -> OID*
    - *Length (in this case) = 9 bytes*
    - *ifSpeed = 0x06092C060102010202020108*
- AMP Uses MIDS (Managed IDs)
  - MIDS encapsulate OIDs (less <type> field)
  - Option to compress OID
  - Makes easy to interoperate with SNMP



- iso (1)
  - org (3)
    - dod (6)
      - internet (1)
        - directory (1)
        - mgmt (2)
          - mib-2 (1)
            - system (1)
            - interfaces (2)
              - ifNumber (1)
              - ifTable (2)
                - ifEntry (1)
                  - ifIndex (1)
                  - ifDescr (2)
                  - ifType (3)
                  - ifMtu (4)
                  - ifSpeed (5)
                  - ifPhysAddress (6)
                  - ifAdminStatus (7)
                  - ifOperStatus (8)
                  - ifLastChange (9)
                  - ifInOctets (10)
                  - ifInUcastPkts (11)
                  - ifInNUcastPkts (12)
                  - ifInDiscards (13)
                  - ifInErrors (14)

OID Types (1/2)

- Full OID
  - Length + Octets
  - Not interpreted by AMP. Used as a unique bitstream.
  - Encoded in ASN.1 BER for now, assuming SNMP Type 6.

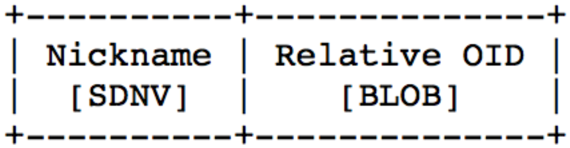
```
                    +----------+
                    | Full OID |
                    | [BLOB]   |
                    +----++----+
                         ||
                         ||
 _______________________/  \_______________________
/                                                  \
+------------+---------+---------+       +---------+
| OID Length | Octet 1 | Octet 2 |  ...  | Octet N |
|   [SDNV]   | [BYTE]  | [BYTE]  |       | [BYTE]  |
+------------+---------+---------+       +---------+
```

- Parameterized OID
  - Full OID followed by AMP Data Collection (DC).
  - DC is a count followed by a series of TLV.
    - *Time, Length, Value*
    - *Type is data type (string, int)*

```
      +----------+------------+
      | FULL OID | Parameters |
      |  [BLOB]  |    [DC]    |
      +----------+-----++-----+
                       ||
                       ||
 _____________________/  \_______________________
/                                                \
+----------+--------+--------+       +--------+
| # Params | Parm 1 | Parm 2 |       | Parm N |
|  [SDNV]  | [BLOB] | [BLOB] |  ...  | [BLOB] |
+----------+--------+--------+       +--------+
```

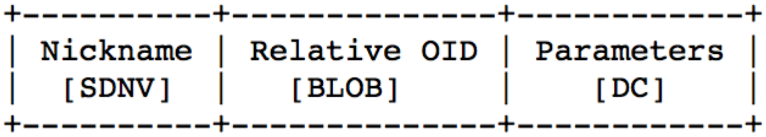
OID Types (2/2)

- Compressed OID
  - AMP supports managed registry of common OID sets.
    - *OIDs can be very long and the portion up to your relative subtree can be reused a lot.*
  - Nickname is an integer that maps to a well-known node in an OID tree.
    - *Relative OID is subtree rooted at that node.*



```
+-----------+---------------+
| Nickname  | Relative OID  |
|  [SDNV]   |    [BLOB]     |
+-----------+---------------+
```

- Compressed, Parameterized OID
  - Compressed OID followed by a Data Collection of Parameters
  - Very similar to a Parameterized OID



```
+-----------+---------------+-------------+
| Nickname  | Relative OID  | Parameters  |
|  [SDNV]   |    [BLOB]     |    [DC]     |
+-----------+---------------+-------------+
```

Application Data Model

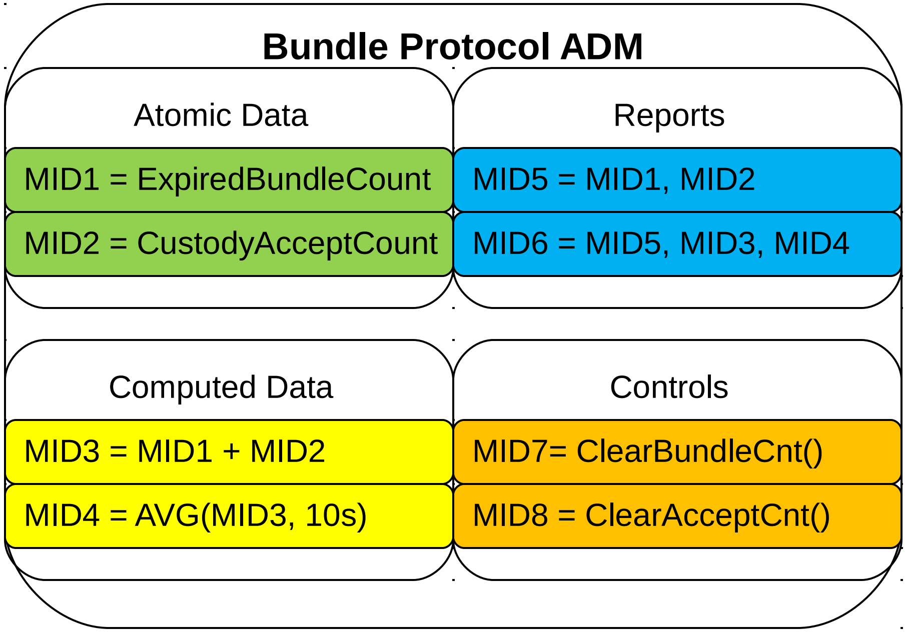
AMP ADMs capture all necessary information for each supported application or protocol..

- Atomic Data and Controls.
  - What immutable data definitions are given for any manager/agent supporting a particular application?
  - What common actions can be taken to manage this application?
- Literals and Operators.
  - What constants are defined for this application?
  - What special operators can be used to compute new data definitions?
- Computed Data.
  - What data definitions are pre-derived from other data definitions?
- Collections.
  - What pre-defined collections of data values (reports) and control sequences (macros) have been created?

ADM Example (1)

Pre-defined data, reports, and controls for applications managed by AMP.

- Pre-defined, atomic data
  - Definitions from MIBs
    - *Global, unique OIDs*
    - *No tag/issuer fields*
    - *All data and reports*
  - Build blocks for user content
    - *Data MIDs can be used in user definitions*
- Pre-defined controls
  - Also global, unique OIDs
  - Opcodes, description, params
  - Build blocks for macro commands
    - *No ability for user-defined controls outside of these pre-defined functions.*



Bundle Protocol ADM

Atomic Data

MID1 = ExpiredBundleCount

MID2 = CustodyAcceptCount

Reports

MID5 = MID1, MID2

MID6 = MID5, MID3, MID4

Computed Data

MID3 = MID1 + MID2

MID4 = AVG(MID3, 10s)

Controls

MID7= ClearBundleCnt()

MID8 = ClearAcceptCnt()

ADM Example (2)

A sample ADM for an application implementing a stack.

“STACK” Application Data Model

Atomic Controls	Computed Data	Atomic Data
- PUSH(X) - POP(X)	- Average POPs	- Stack Depth - Total Items - Total # POPs

Literals	Data Collections	Control Collections
- MAX_DEPTH = 10	Report 1: - Cur. Stack Depth - Total Items - Average POPs	EMPTY : Stack Depth > 0 POP(X)

AMP Agent ADM

From draft-birrane-dtn-adm-agent-00

- Captures all behavior of an AMP Agent
  - Keeps AMP functional specification simple
  - Items available to AMA/AMP ecosystem because this ADM must be implemented by any deployed AMP agent.
- Primitive Values
  - Counters, number of AMP types created, active, etc…
- Reports
  - Full report definitions. Users may customize their own.
- Controls
  - All functions to create, update, delete, and other wise manage reports, rules, macros, and other AMA types.
- Operators
  - Full math function spec
    - *+, -, *, /, %, ^, &, |, &&, ||, !, abs(), <, >, <=, >=, !=, ==, >>, <<*

Thank you!

Questions?