

Asynchronous Management

IETF 100

Edward Birrane
Edward.Birrane@jhuapl.edu
443-778-7423



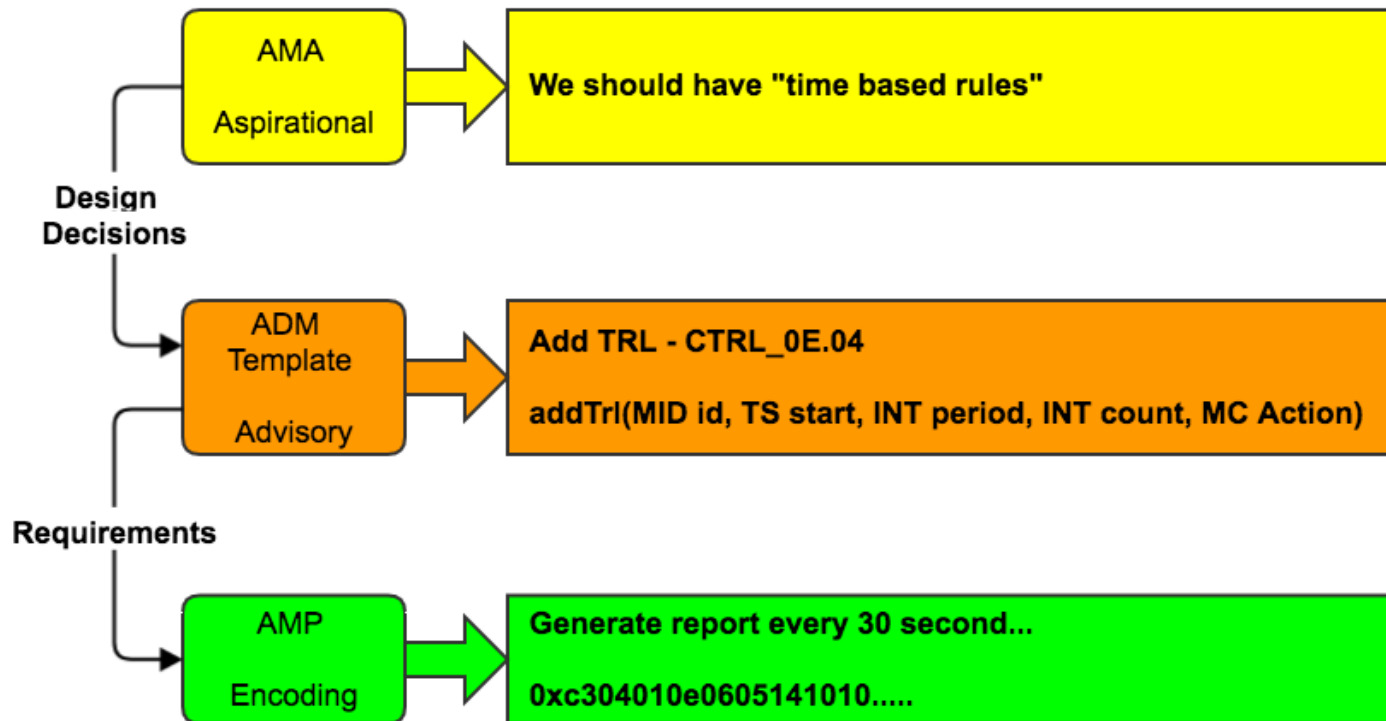
APL

JOHNS HOPKINS UNIVERSITY
Applied Physics Laboratory

High-Level Overview

- **Asynchronous Management Architecture (AMA)**
 - Proposed architecture for delay-tolerant network management.
 - draft-birrane-dtn-ama-06 latest draft.
 - *Fixed misspellings.*
 - *Added paragraph clarifying support for concept of tabular data.*
- **AMA Application Data Modeling (ADM)**
 - Proposed data model compliant with the AMA.
 - **draft-birrane-dtn-adm-00**
 - *YANG schemas for data model*
 - *JSON examples of populated models for a trivial application*

AMA/ADM/AMP Interactions



AMA: Overview

From draft-birrane-dtn-ama-06

■ 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

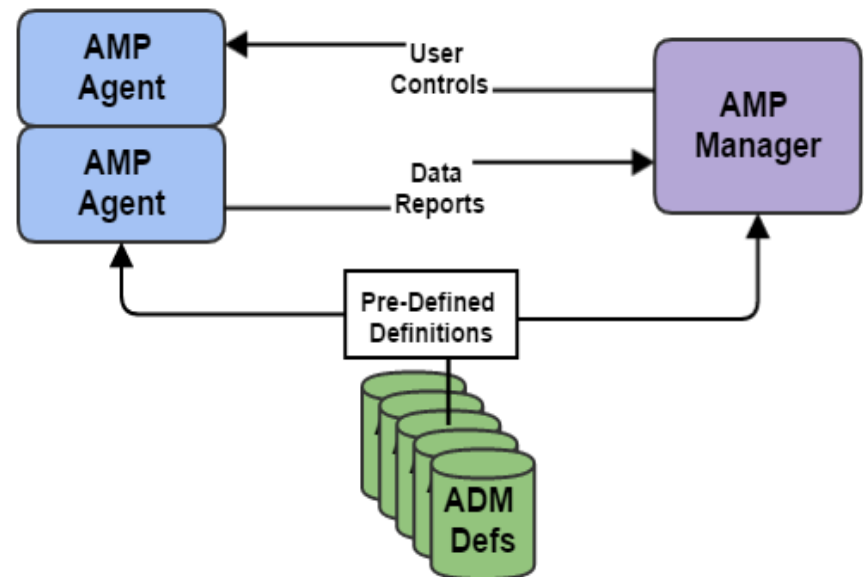
- **Intelligent Information Push:** Can't rely on others.
- **Minimize Message Size:** Increase probability of delivery.
- **Absolute Data Identification:** Pre-shared, global naming.
- **Custom Data Definition:** Send minimal necessary data sets.
- **Autonomous Operation:** Decisions local to Agent



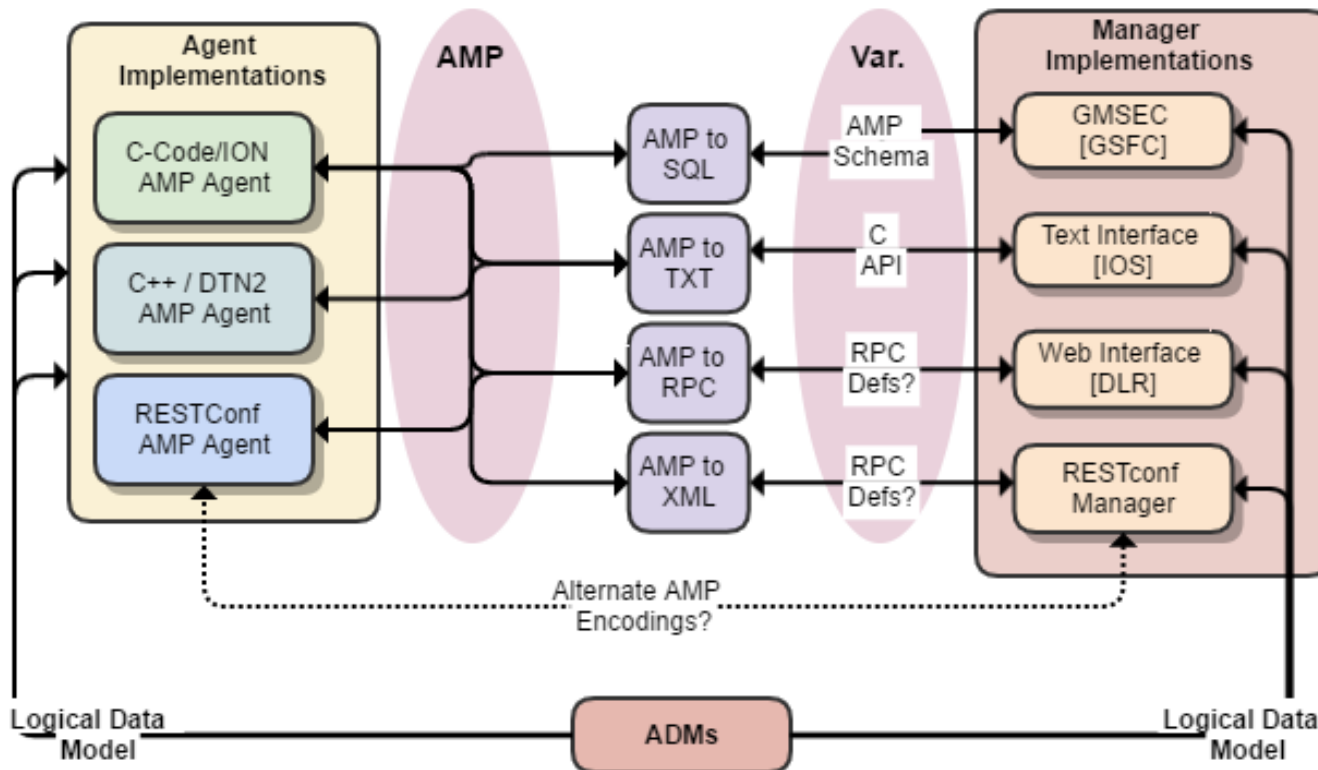
AMA: The Simple System Model

From draft-birrane-dtn-ama-06

- **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
 - Schemas in YANG
 - Preconfiguration reduces msg size



AMA: The Actual System Model



ADM Template: Logical Modeling

- Separate the data specification from its encoding.
 - Use AMP specification to define how to compactly encode ADM items
- ADMs Schemas will define logical models
 - Designed to identify minimum set of information per data model
 - Remove any “encoding hints” from the models.
 - Use the YANG modelling language
 - *Tools exist to validate YANG schemas for correctness and plot dependencies.*
- ADMs can be defined in JSON using JSON encodings for YANG schemas
 - Conventions will be defined to make JSON writing expressive and “easy”
 - Reuse existing notations/delimiters where possible (query string)
- Define compilers/adapters
 - Presuppose adapters/compilers to generate encodings as necessary

ADM Template: Logical Data Model

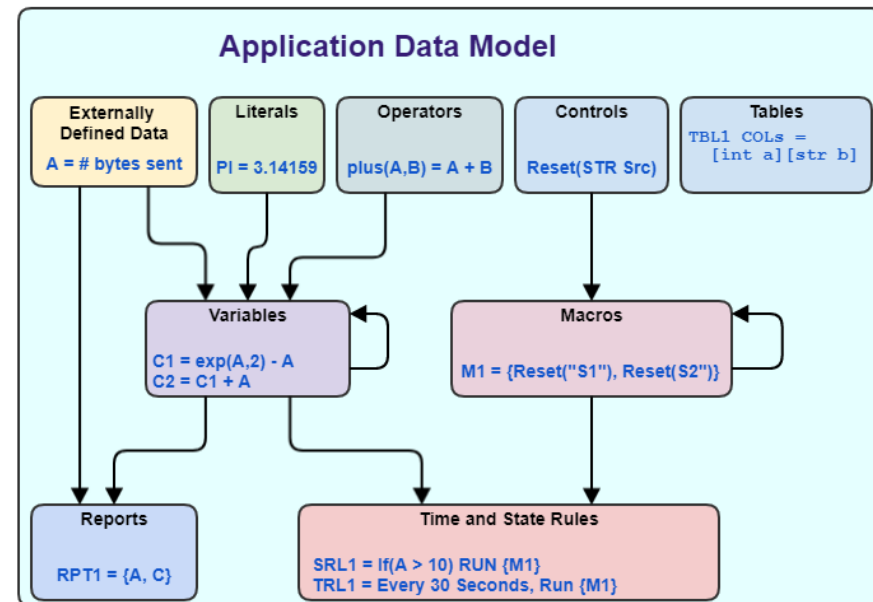
Static Elements

- Solely defined by their ADM.
- EDDs: collected by agents.
- Literals: useful constants.
- Ops: opcodes for math functions.
- Ctrls: opcodes for agent behavior.

Dynamic Elements

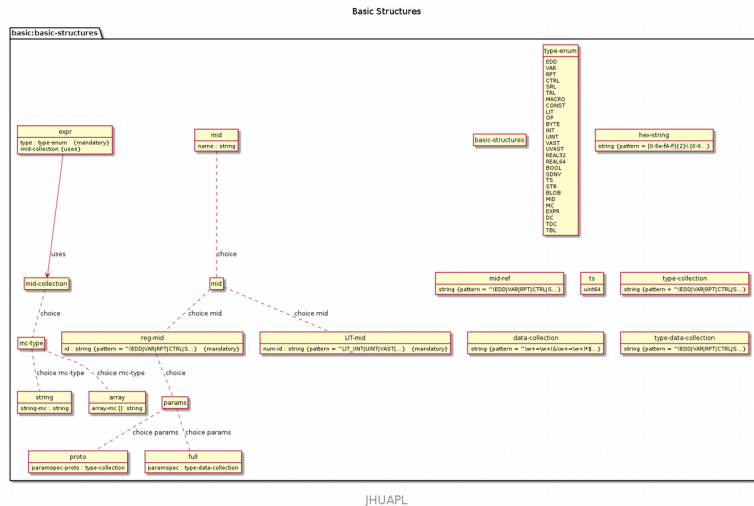
- Defined by ADM or by User
- ADM definitions are immutable.
- Vars: strong-typed variables, including a type for “expression”.
- Macro: Ordered set of Ctrls.
- Rpts: Ordered sets of data
- Rules: Time or State based autonomy.

An ADM defines 9 types of data for each application/protocol managed in the AMA.



ADM Template: YANG Basic Structures

- **Types**
 - Enumerations, typedefs, etc...
- **Basic Structures**
 - MID – Common Identifier
 - Structure identifiers (parameterized or not)
 - Literal Identifiers
 - EXPR -- Expressions
 - COLLECTIONS – Arrays of things
 - MID Collections
 - Type Collections (e.g. function definition)
 - Data Collections
 - Typed Data Collections (e.g. function calls)
 - TIMESTAMP



Tools exist to validate YANG schemas for correctness and plot dependencies.

ADM Template: YANG Complex Structures

- **Complex Structures**
 - CONSTANTS – PI = 3.14159
 - CONTROLS – Parameterized command opcodes
 - STATE RULES – Condition -> Action
 - TIME RULES – Period -> Action
 - TABLES – Bulk reporting
 - VARIABLE – Strong-Typed, User-Defined
 - EXTERNALLY DEFINED DATA – Firmware sampled counts (# bundles sent)
 - OPERATOR – Typed algebraic expressions
 - REPORT TEMPLATE – Templated data return

Complex Structures

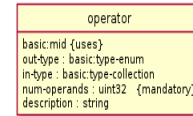
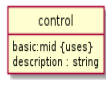
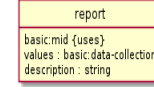
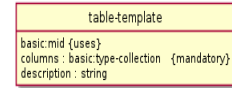
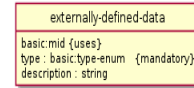
Complex Structures

basic:basic-structures

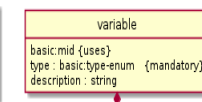
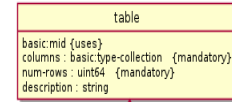
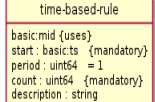
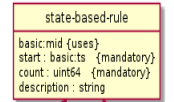
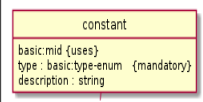


complex_complex_structures

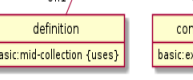
complex-structures



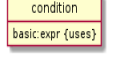
complex:complex-structures



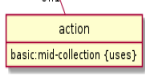
value



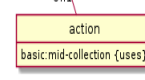
0..1



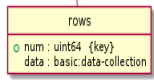
0..1



0..1



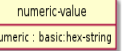
0..1



0..N



0..1



choice

choice value

choice value

JHUAPL



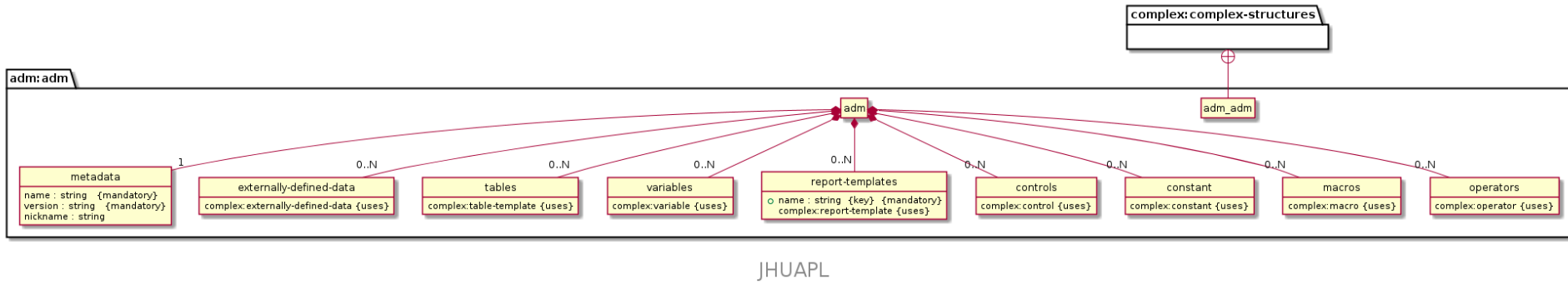
ADM Template: Scope

- Things that are solely defined in the ADM
 - Metadata – Name, Version
 - EDDs – *All* external data definitions.
 - TABLEs – *All* table definitions.
 - CONTROLS – *All* control definitions
 - CONSTANT– *All* constant definitions
 - OPERATORS -- *All* operator definitions
- Things that are defined in the ADM and in networks
 - VARIABLEs – *Some* variable definitions.
 - MACROs -- *Some* macro definitions
 - REPORT TEMPLATES – *Some* Report definitions
- Things not identified in the ADM, in networks only
 - TIME RULES
 - STATE RULES



ADM Template: YANG Schema

Application Data Model



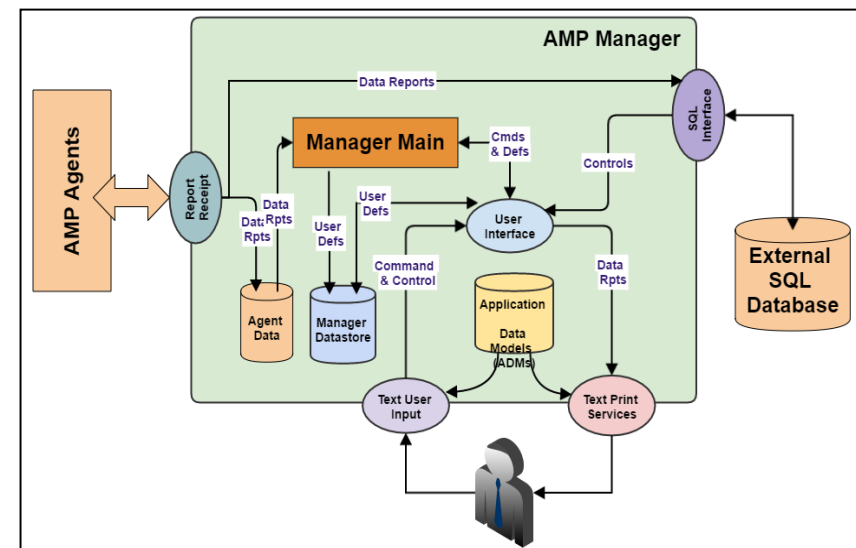
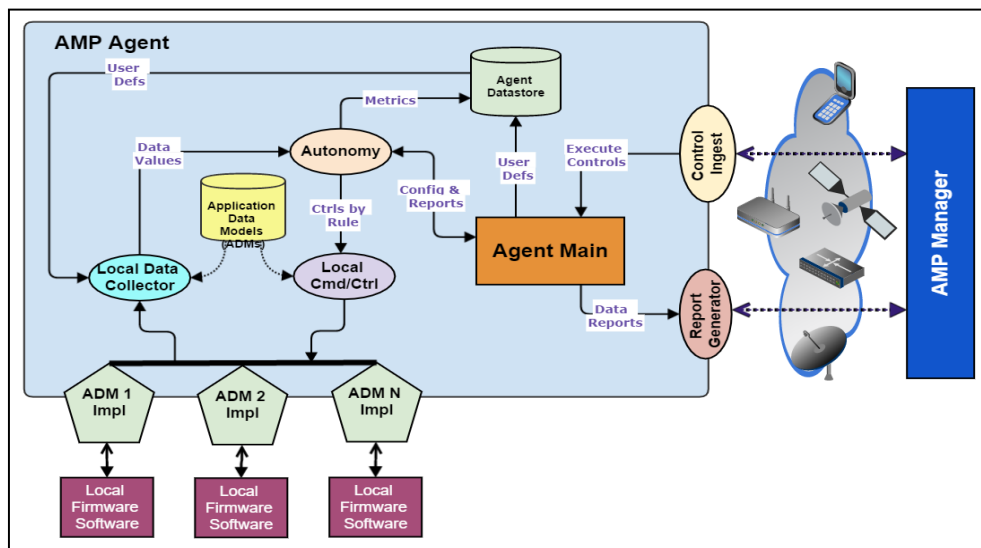
ADM Template: JSON Ipnadmin example

```
"adm:metadata": {
  "name": "Interplanetary internet (IPN) scheme", "version": "V0.0",
  "nickname": "60 IPN Metadata, 61 IPN EDDs, 62 IPN Variables, 63 IPN Report Templates, 64 IPN Controls, 65 IPN Constants,..."
},
"adm:externally-defined-data": [
  {
    "name": "version", "id": "EDD_00.61", "type": "STR",
    "description": "This is the version of ion currently installed."
  }
],
"adm:tables": [
  {
    "name": "exitRules", "id": "TBL_00.68", "columns": "UINT:firstNodeNbr&UINT:lastNodeNbr&STR:qualifier&STR:gatewayEndpointId",
    "description": "lists all exit rules"
  }
],
"adm:controls": [
  {
    "name": "exitAdd", "id": "CTRL_00.64", "paramspec-proto": "UINT:firstNodeNbr&UINT:lastNodeNbr&STR:gatewayEndpointId",
    "description": "This control establishes an \"exit\" for static default routing."
  }
]
```



Asynchronous Management Protocol (AMP)

- Encoding of the ADM in the context of a protocol
 - Current prototyping efforts answer the questions
 - *“Is the data model unambiguous and implementable”*
 - *“What is representative performance on representative platforms”*
 - Reference implementation maturing in ION (3.6.x releases)





Questions?



APL

