AMA Data Model

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

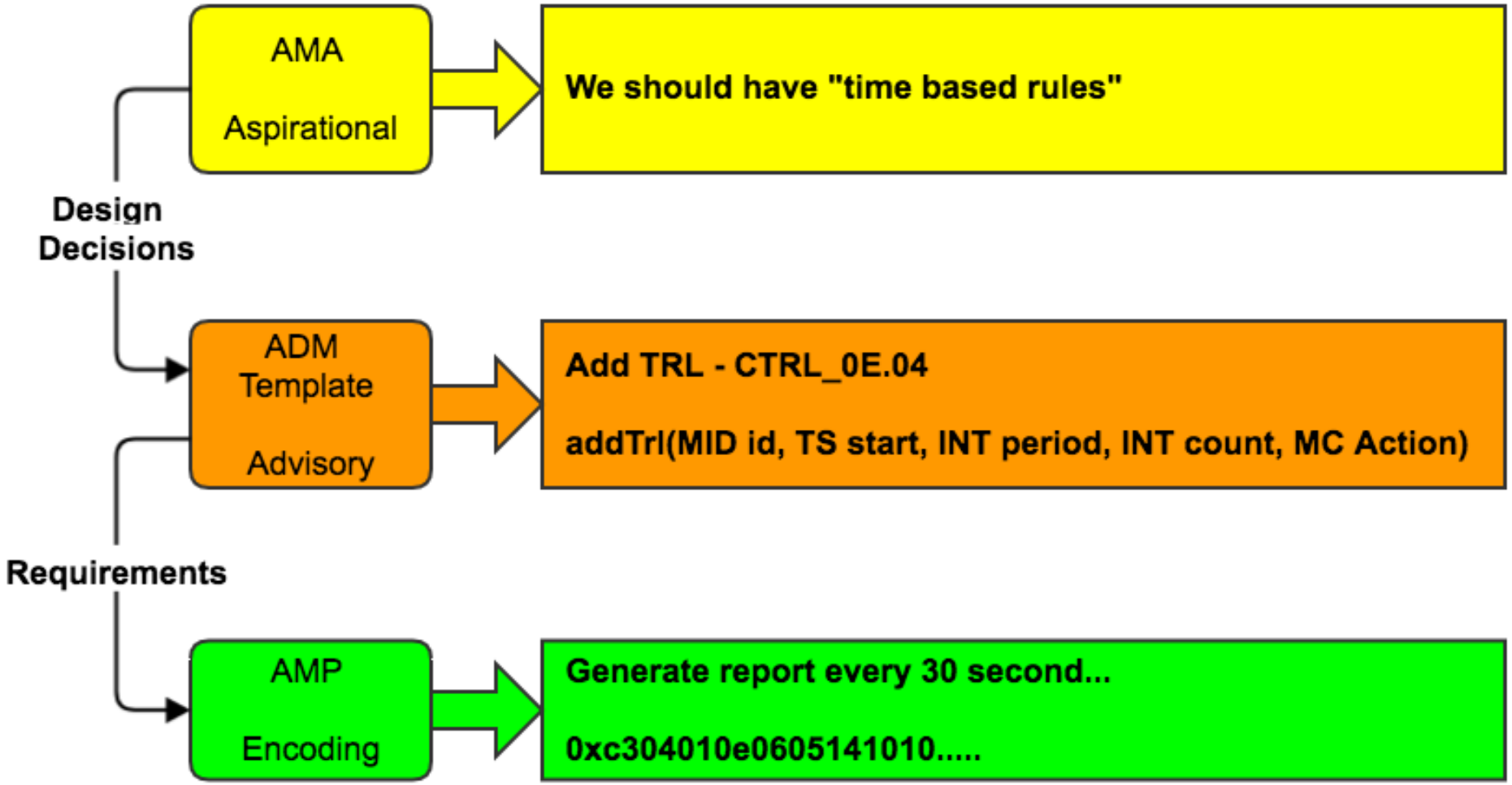
JOHNS HOPKINS UNIVERSITY
Applied Physics Laboratory

We need a compact, efficient autonomy model for DTN network management

Automated network management being “discovered” for terrestrial applications.

- Automation and autonomy is more than configuration
  - Historical NM protocols cannot work in DTN environments
  - Some issues with emerging thoughts on automated NM in other areas
    - *Too verbose: Layering automation over existing protocols that don’t work in DTN will not help the DTN use case.*
    - *Insufficient autonomy: summarizing data sets is not enough.*
    - *Too synchronous: Relying on omniscient operators in the network doesn’t work for DTN use cases.*
- An autonomy model and efficient encoding is needed for DTNs
  - AMA provides a rationale for this.
  - No changes to AMA since last IETF.
  - ADM attempts to formalize the AMA data/autonomy model

AMA/ADM/AMP Interactions



ADM Template: Logical Modeling

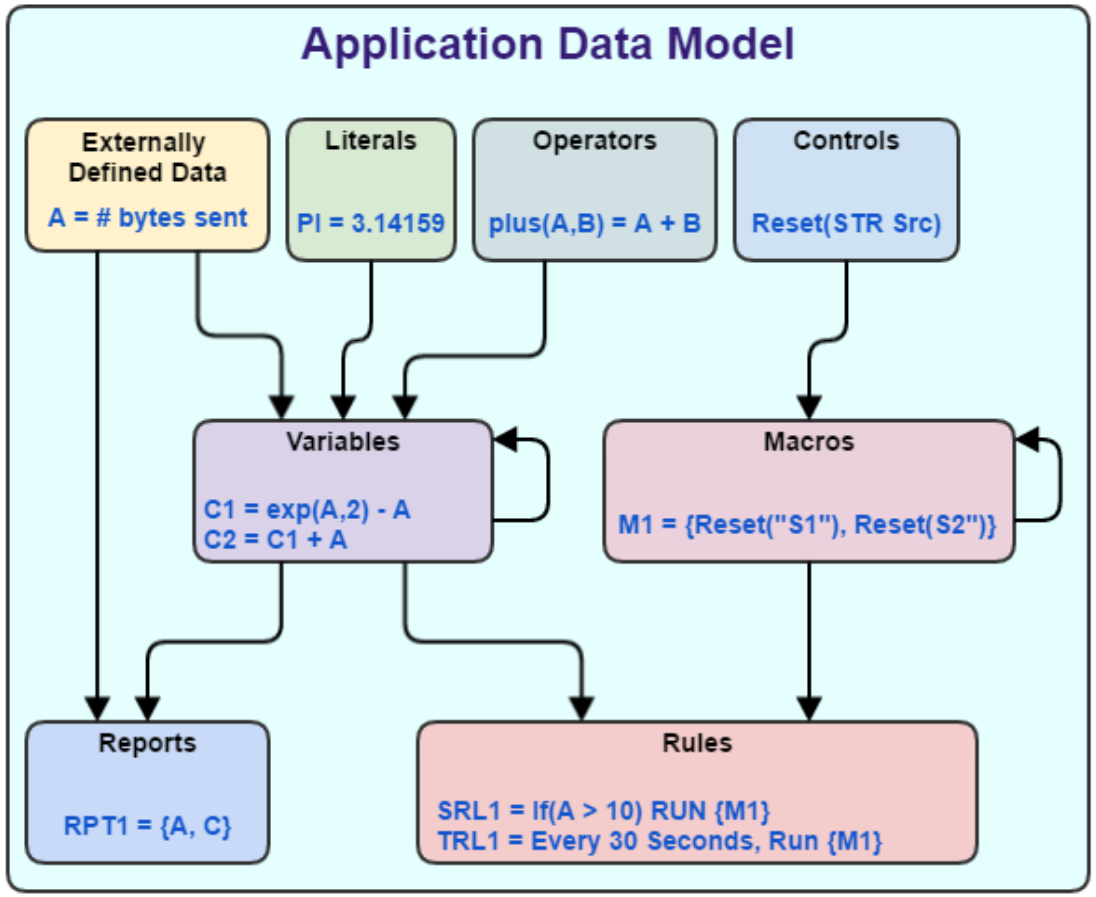
- Separate the data specification from its encoding.
  - Use AMP specification to define how to compactly encode ADM items
- ADMs Schemas will define logical models
  - Designed to identify minimum set of information per data model
  - Remove any “encoding hints” from the models.
  - Use the YANG modelling language
    - *Tools exist to validate YANG schemas for correctness and plot dependencies.*
- ADMs will be defined in JSON
  - 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

- "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 defines 8 types of data for each application/protocol managed in the AMA.



ADM Template Status

- Candidate JSON syntax proposed – in SME review
  - Examples in following slides
- ION Administrative functions ported to JSON ADMs
  - ION BP Admin (bpadmin)
  - ION Admin (ionadmin)
  - ION IONSEC Admin (ionsecadmin)
  - ION IPN ADMIN (ipnadmin)
  - ION LTP ADMIN (ltpadmin)
- Standard ADMs ported
  - BP Agent
  - LTP Agent
  - BPSEC Agent

Example: Externally Defined Data

- Basic type

"**name**": "num_good_tx_bcb_blks_total",
"**type**": "UINT",
"**description**": ”All successfully Tx BCB blocks”

- Parameterized data type

"**name**": "num_good_tx_bcb_blks_src",
"**type**": "UINT",
"**parmspec**": [{"STR":"Src"}],
"**description**": "Successfully Tx BCB blocks from SRC”

Example: Variables

- Variables Example

"**name**": "total_bad_tx_blks",
"**type**": "UINT",
"**initializer**": {
"**type**": "UINT",
"**postfix-expr**": ["EDD.item1('0')","EDD.item2('1')","OP.+UINT"]
},

"**description**": "# total items (# item1 + # item2)."

Example: Tables and Reports

- Table Example

"**name**":"keys",
"**columns**": [{"STR":"ciphersuite_names"}],
"**description**": "This table lists supported ciphersuites."

- Report Example

"**name**": "full_report",
"**parmspec**": [{"STR":"Source"}],
"**definition**" : [
"EDD.data_item1",
"EDD.data_item2('1')",
"EDD.data_item3(Source)",
"EDD.data_item4('1', Source)",
],
"**description**": "A full report."

Example: Controls and Macros

- Control Example

"**name**": "reset_src_cnts",
"**parmspec**": [{"STR":"src"}],
"**description**": "This control resets counts for the given source.”

- Macro Example

"**name**": "user_list",
"**definition**": [
"CTRL.list_vars",
"CTRL.list_rptts”
],
"**description**": "List user defined data."

Example: Constants and Operators

- Constant Example

"**name**": "PI",
"**type**": "FLOAT",
"**value**": 3.14159,
"**description**": "The value of PI."

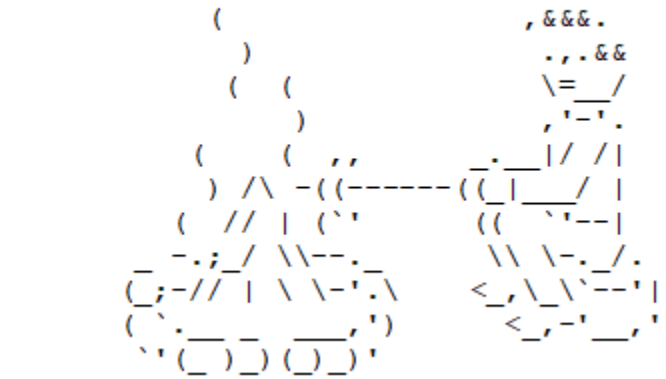
- Operator Example

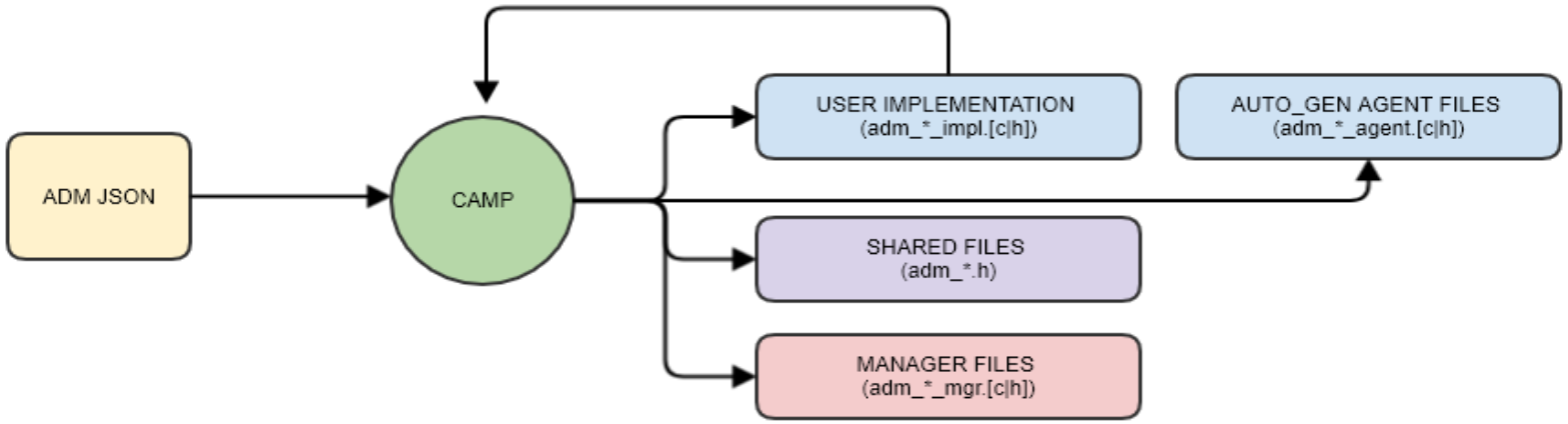
"**name**": "+INT",
"**result-type**": "INT",
"**in-type**": ["INT", "INT"],
"**description**": "Int32 addition"

ADM Auto Generation

- ION C-generating AMP Python Script (CampPython)
  - Github project
    - *C API for AMP defined for ION (3.x or 4.x line)*
    - *Produces .c/.h files per ADM*
    - *Includes a user-customizable .c/.h) and “round-tripping”*
    - *camp <adm.json> -c <old_impl.c> -s <old_impl.h>*
  - Adding and maintaining ADMs much simplified
    - *But, now, lots of data. 200+ data items, ~100 controls/operators*

```
# CAmpPython
```





CAMP User Implementation File Example

```
value_t adm_bpsec_get_ciphersuite_names(tdc_t params)
{
    value_t result;
   /* +--------------------------------------------------------------+
    * |START CUSTOM FUNCTION get_ciphersuite_names BODY
    * +--------------------------------------------------------------+/

    char *tmp = bpsec_instr_get_csnames();
    result.value.as_ptr = STAKE(size));
    memcpy(result.value.as_ptr, tmp, size);
    SRELEASE(tmp);
    result.type = AMP_TYPE_STRING;

   /* +--------------------------------------------------------------+
    * |STOP CUSTOM FUNCTION get_ciphersuite_names BODY
    * +--------------------------------------------------------------+/
    return result;
}
```

APL

Outstanding Issues

- AMA
  - Diverse review. To date, consensus.
  - At last WG it was decided by chairs to ask to adopt AMA as a WG document.
- ADM Template
  - Initial publish of JSON and YANG.
  - Some pushback on separating YANG and NETCONF
  - Specify JSON format for everyone.
  - YANG interface for NETCONF users.
- ADMs
  - Drafts of all ADMs now in JSON. No more hand-maintaining hex values.
- AMP
  - In progress. CBOR. ACL. Remove data model (now in ADM template)