Delay-Tolerant Networking   E. Birrane
Internet-Draft   E. DiPietro
Intended status: Informational   D. Linko
Expires: January 3, 2019   Johns Hopkins Applied Physics Laboratory
                                                              M. Sinkiat
                                      ASRC Space And Defense, NASA GSFC
                                                           July 2, 2018

AMP Manager SQL Interface
draft-birrane-dtn-ampmgr-sql-01

Abstract

This document describes a proposed public interface through which an application, such as a network management console, interacts with an Asynchronous Management Protocol (AMP) Manager via a database supporting the Structured Query Language (SQL). The use of SQL as an interfacing layer provides a natural way to describe interactions with an AMP Manager independent of a particular implementation of either the Manager or the application. Specifically, this document presents a database schema capturing how to send controls to a Manager and how to accept reports received by a Manager from one or more AMP Agents.

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 https://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 January 3, 2019.

Table of Contents

1. Introduction 3
1.1. Purpose 3
1.2. Scope 4
1.3. Requirements Language 4
2. Database Overview 4
3. Constants 6
3.1. Data Types - data_types 6
3.2. Incoming State - incoming_state 6
3.3. Outgoing State - outgoing_state 8
4. ADM Support 9
4.1. ADM Nicknames - adm_nickname 9
4.2. Supported ADMs - adm 10
4.3. ARIs 11
4.3.1. Individual ARI Parameter - parmspec 11
4.3.2. ARI Parameter Collection - parmspec_set 12
4.3.3. ARI - ari 12
4.3.4. ARI - ari_definition 13
5. Agent Support 14
5.1. Registered Agents - registered_agents 14
6. ARI Information 15
6.1. Data Collections 15
6.1.1. Data Collection Entry - data_value 15
6.1.2. Data Collection - data_set 17
6.2. ARI Collections 17
6.2.1. ARI Collection Entry - ac 17
6.2.2. ARI Collection - ac_set 18
7. Operator Support 18
7.1. In Type Set - in_type_set 18
7.2. In Type - in_type 19
8. Outgoing Message Support 19
8.1. Outgoing Messages - outgoing_message 20
8.2. Outgoing Message Groups - outgoing_message_group 20
9. Incoming Message Support 21
9.1. Incoming Messages - incoming_messages 22
9.2. Incoming Message Groups - incoming_message_group 22
10. Compound Data Types 23

10.1. TNVC Support 23
10.1.1. Type Name Value Collection - tnvc 23
10.1.2. Type Name Value Collection Set - tnvc_set 24
10.2. Expression Support 24
10.2.1. Expression Set - expression_set 24
10.2.2. Individual Expression - expression 25
11. IANA Considerations 25
12. Security Considerations 25
13. References 25
13.1. Informative References 25
13.2. Normative References 26
Appendix A. Acknowledgements 26
Authors' Addresses 26

1. Introduction

This document presents a public interface through which an application, such as a network management console, interacts with an Asynchronous Management Protocol ([I-D.birrane-dtn-amp]) Manager via a database supporting the Structured Query Language (SQL). Such an interface is useful as an implementation independent way of specifying how an application may interact with an AMP Manager to issue commands (such as through a custom graphical user interface) and to receive reports (as they are received by one or more AMP Agents).

1.1. Purpose

This document describes a database layout comprised of a series of names tables and the columns the comprise those tables. Where appropriate, primary and foreign key constraints are also discussed. This set of tables presents a data model through which all AMP Manager roles and responsibilities, as defined in the Asynchronous Management Architecture ([I-D.birrane-dtn-ama]), can be accomplished.

Application developers can use this specification to describe how to populate a database with AMP-related information such that an AMP Manager implementation can read and use this data to effect AMP behavior. By reading and writing the tables in accordance with this specification, applications can claim conformance with the AMP Manager regardless of which AMP Manager implementation is used, so long as such a Manager is also in conformance with this specification.

AMP Manager developers use this specification to describe how AMP users input actions to the Manager and how to send received reports back to those users.

1.2. Scope

This document covers table names and the names, data types, default values, and comments associated with each column of each names table. These types should be appropriate for any database implementing a SQL interface and SHOULD NOT use any language or function specific to a particular SQL database vendor.

This document does not specify the setup, configuration, administration, or other function associated with a particular SQL database vendor. Further, this document does not specify how either the application or the AMP Manager log on to the database, or how database communications are verified and secured. Finally, this document does not discuss the architecture associated with incorporating a database between an application and an AMP Manager, as such architectures are likely tightly coupled to a network deployment.

1.3. 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. Database Overview

This specification assumes that all tables exist in a database called "amp_core", capturing the information necessary to capture core operations associated with the AMP Manager.

The schema contains tables capturing information about various areas of the AMP Manager interface. The following table types are defined.

- Constants - These tables contain those constants defined in the AMP and AMA specifications as well as certain constants that are defined in this specification.

- ADM Support - These tables capture all information that is solely defined in an ADM. Outside of adding support for a new ADM, the data in these tables should never be changed.

- AMP Agent Support - These tables capture information about Agents in the system.

- Compound Data Types Support- These tables capture information about TNVCs and expressions.

- Operator Support- These tables capture the information that is needed to utilize operators.

- ARI Information - These tables capture all information used to define Managed Identifiers (ARIs) for the Manager. This includes ARI definitions from ADMs custom definitions from users in the system. These tables also capture parameters, object identifiers, and ARI collections.

- Outgoing Support - These tables capture information provided by an application to a Manager to be sent out to an AMP agent.

- Incoming Support - These tables capture information provided by an Agent and incoming to the Manager to be passed along to the application.

The specification comprises tables, as in Figure 1.

Table Name	Table Type
data_type	Constants
incoming_state	Constants
outgoing_state	Constants
tnvc_set	Compound Data Type
tnvc	Compound Data Type
expression_set	Compound Data Type
expression	Compound Data Type
in_type_set	Operator Support
in_type	Operator Support
adm_nickname	ADM Support
adm	ADM Support
parmspec	ADM Support
parmspec_set	ADM Support
registered_agents	Agent Support
data	ARI Information
data_set	ARI Information
ac	ARI Information
ac_set	ARI Information
ari	ARI Information
ari_definition	ARI Information
outgoing_message_group	Outgoing Support
outgoing_message	Outgoing Support
incoming_message_group	Incoming Support
incoming_message	Incoming Support

Figure 1: amp_core tables

3. Constants

3.1. Data Types - data_types

Data types, as defined in the AMP, enumerate the types of information associated with collections of data, such as defined in ARI parameters, computed values, and report definitions.

The format of the table is as follows.

Field	Type	Null	Key	Default	Extra
enumeration	int(10) unsigned	NO	PRI		
name	varchar(50)	NO	UNI		
description	varchar(255)	NO		""	

data_types

enumeration
: The primary key for this table, and the enumerated value of the data type from the AMP specification.

Name
: The name associated with this data type.

Description
: The description associated with this data type.

An example of such a table is illustrated in [I-D.birrane-dtn-adm].

3.2. Incoming State - incoming_state

When reports are being received by a Manager from an Agent, they will be written into various Incoming Support table types. However, the application reviewing these incoming reports should not start to read them until the Manager has finished receiving and persisting them into the database.

The incoming_state table identifies three different wait states associated with receiving reports from Agents. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
state_id	tinyint(3) unsigned	NO	PRI		
name	varchar(50)	NO			
description	varchar(255)	NO			

incoming_state

state_id
: The primary key for this table, and the enumerated value of the incoming state. Three states are defined in this specification, as follows.

  0 - Initializing
  : This state signifies that a Manager is receiving a set of information and rows associated with this state should not be read by an application.

  1 - Ready
  : This state signifies that a Manager has completed receiving reports and that rows associated with this state may be processed by an application.

  2 - Processed
  : This state signifies that an application has completed processing reports and that either a Manager or an application can remove rows associated with this state at any time.

Name
: The name associated with the incoming state (Initializing, Ready, Processed).

Description
: The description associated with the incoming state.

An example of such a table is illustrated below.

ID	Name	Description
0	Initializing	Manager is receiving reports.
1	Ready	Manager has completed reception.
2	Processed	Application is done processing reports.

incoming_state example

3.3. Outgoing State - outgoing_state

When controls are being sent via a Manager to an Agent, they will be written into various Outgoing Support table types. However, the Manager receiving these outgoing controls should not start to read them until the application has finished writing them into the database.

The outgoing_state table identifies four different wait states associated with sending controls to Agents. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
state_id	tinyint(4)	NO	PRI		
name	varchar(50)	NO			
description	varchar(255)	NO			

outgoing_state

state_id
: The primary key for this table, and the enumerated value of the outgoing state. Four states are defined in this specification, as follows.

  0 - Initializing
  : This state signifies that an application is preparing a set of controls and rows associated with this state should not be read by a Manager.

  1 - Ready
  : This state signifies that an application has completed preparing the controls and that rows associated with this state may be processed by the Manager for sending to one or more Agents.

2 - Processing
: This state signifies that the Manager is in the process of sending associated controls to one or more Agents. Rows in this state should not be modified by an application as it could affect the controls sent by the Manager.

3 - Sent
: This state signifies that the Manager has completed sending the set of controls and that either a Manager or an application can remove rows associated with this state at any time.

name
: The name associated with the outgoing state (Initializing, Ready, Processing, Sent).

description
: The description associated with the outgoing state.

An example of such a table is illustrated below.

state_id	name	description
0	Initializing	Application writing controls.
1	Ready	Ready for Sending to Agent.
2	Processing	Manager sending controls.
3	Sent	Manager send completed.

outgoing_state Example

4. ADM Support

4.1. ADM Nicknames - adm_nickname

The adm_nicknames table identifies all of the nicknames associated with supported ADMs. A Nickname is the compression of a shared portion of an ARI. Nickname enumerations MUST be unique. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
adm_id	int(10) unsigned	NO	MUL		
nickname_enum	int(10) unsigned	NO			
label	varchar(255)	NO			

adm_nicknames

adm_id
: The primary key for this table.

nickname_enum
: The enumeration of the nickname.

label
: This is the label of the nickname.

4.2. Supported ADMs - adm

The adm table identifies all of the ADMs supported by the AMP Manager and associated application. The format of the table is as follows.

Field	Type	Null	Key	Default	Extra
adm_id	int(10) unsigned	NO	PRI		auto_increment
name	varchar(255)	NO	UNI		
version	varchar(255)	NO			
description	varchar(255)	NO			
Organization	varchar(255)	NO			

adm

adm_id
: The primary key for this table.

name
: The name of the supported ADM.

version
: The string representing the version of the ADM. A string is used to allow for a variety of version formats.

description
: This is the description of the ADM.

Organization
: This is the issuing organization of the ADM.

4.3. ARIs

ARIs identifying items such as Controls may accept parameters to customize their behavior. When defined in the context of an ADM, a parameterized ARI only includes the non-parameterized portion of the ARI followed by the expected data types for the parameterized portion of the ARI. This, essentially, acts as a template for populating a specific instance of the ARI with actual data.

This "template" is referred to as a ARI, as it is used to generate ARI instances. The instances of an ARI are called ARI definitions. The amp_core database schema identifies three tables used to capture ARI definitions from ADMs: parmspec, parmspec_set, ari, and ari_definition.

4.3.1. Individual ARI Parameter - parmspec

The parmspec table contains a row for each parameter associated with an ARI. All of the parameters for an ARI, together, are considered a "collection" of parameters. The format of the table is as follows.

Field	Type	Null	Key	Default	Extra
name	varchar(45)	NO			
parm_id	int(10) unsigned	YES	MUL	NULL	
parm_order	int(10) unsigned	NO	UNI	0	
type_id	int(10) unsigned	YES	MUL	NULL	

parmspec

name
: The name of the parmspec collection.

parm_id
: The id of the collection for this parameter. A parameter collection is the ordered set of parameters that describe an ARI. This is a foreign key that corresponds to the parm_id of the parmspec_set table.

parm_order
: The 0-based ordering of this parameter within the collection.

type_id

The type of this parameter. This must be one of the known AMP types and, as such, is a foreign key that corresponds to the enumeration of the data_type table.

4.3.2. ARI Parameter Collection - parmspec_set

The parmspec_set table represents the ordered set of parameters associated with an ARI. The format of the table is as follows.

Field	Type	Null	Key	Default	Extra
parm_id	int(10) unsigned	NO	PRI		auto_increment
comment	varchar(255)	NO			

parmspec_set

parm_id
The primary key for this table.

comment
This is the description of the parmspec set.

4.3.3. ARI - ari

The ari table captures the ARIs in the database. Some ARIs will be auto-populated from ADMs. Others will be added dynamically by users of the system. As per [AMP], an ARI without an identified Issuer field is assumed to be as defined in an ADM. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
curr_id	int(10) unsigned	NO	PRI		
name	varchar(50)	NO	UNI	Unnamed	
ari_id	int(10) unsigned	NO	MUL		
data_id	int(10) unsigned	NO	MUL		
description	varchar(255)	YES			

ari

curr_id
The primary key for this table.

name
: The name of the ARI.

ari_id
: This is the id of the ARI. This is a foreign key that corresponds to the ari_id in the ari_definition table.

data_id
: This is a foreign key to the data_set table that corresponds with the data_id of the data_set table.

description
: This is the description of the ari.

4.3.4. ARI - ari_definition

The ari_definition table stores the metadata of all known ARIs. The format of the table is as follows.

Field	Type	Null	Key	Default	Extra
ari_id	int(10) unsigned	NO	PRI		auto_inc
name	varchar(50)	NO	UNI	Unnamed	
type_id	int(10) unsigned	NO	MUL	NULL	
nn_id	int(10) unsigned	YES	MUL	NULL	
ari_bytestring	varchar(255)	NO			
parm_id	int(10) unsigned	YES	MUL	NULL	
issuer	bigint unsigned	YES		NULL	
tag	bigint unsigned	YES		NULL	
structure_id	varchar(255)	YES	MUL	NULL	
in_type_id	int(10) unsigned	YES	MUL	NULL	

ari_definition

ari_id
: The primary key for this table.

nn_id
: The nickname associated with this ARI, if applicable. This is a foreign key that corresponds to the id in the adm_nickname table.

parm_id
: The parameter collection for this ARI. This is a foreign key that corresponds to the parm_id in the parm_set table.

issuer_flag
: A binary value representing whether the ARI has an issuer field (value 1) or not (value 0).

tag_flag
: A binary value representing whether the ARI has a tag field (value 1) or not (value 0).

type_id
: The data type associated with this ARI. This is a foreign key that corresponds to the data_id in the data_types table.

name
: A human-readable name for this ARI.

ari_bytestring
: The enumeration of the ARI.

structure_id
: This is the structure type of the ARI (EDD, VAR,etc.). This is a foreign key that corresponds to the data_id in the data_types table.

in_type_id
: This is the in types that are needed to use a specific operator. This is a foreign key that corresponds to the in_type_id in the in_type_set table.

5. Agent Support

5.1. Registered Agents - registered_agents

The registered_agents table lists the network identifiers for each Agent known in the network. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
registry_id	int(10) unsigned	NO	PRI		auto_increment
agent_id	varchar(128)	NO		ipn:0.0	

registered_agents

registry_id
: The primary key for this table.

agent_id
: This is the identifier for the Agent, suitable for passing into a network send call. A string representation is selected to best capture the identifier format for a particular network.

6. ARI Information

6.1. Data Collections

Data collection tables capture the Data Collection (DC) data type as defined in [I-D.birrane-dtn-amp]. Similar to the parmspec and parmspec_set tables, Data Collections are represented as an ordered collection of individual data items, with one table representing the collection itself, and another table holding the ordered data within the collection.

6.1.1. Data Collection Entry - data_value

The data_value table holds data collection entries, one per row. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
data_id	int(10) unsigned	NO	MUL		
data_order	int(10) unsigned	NO	UNI	0	
data_type	int(10) unsigned	NO	MUL		
ari_id	blob	YES	MUL	NULL	
byte	longblob	YES		NULL	
tnvc_id	int(10) unsigned	NO	MUL		
ac_id	int(10) unsigned	NO	MUL		
vast	bigint	YES		NULL	
uvast	bigint unsigned	YES		NULL	
real	longblob	YES		NULL	
str	varchar(255)	YES		NULL	
bool	boolean	YES		NULL	
ts	datetime	YES		NULL	

data_value

data_id
: The id of the data collection. This is a foreign key that corresponds with the data_id of the data_set table.

data_order

The 0-based order of this entry in the encapsulating collection.

data_type
: The type of this data collection. This is a foreign key that corresponds to the enumeration in the data_types table.

tnvc_id
: This is a foreign key that corresponds to the tnvc_id of the tnvc_set table.

ac_id
: This is a foreign key that corresponds to the ac_id of the ac_set table.

byte
: The byte value shows the value of the data (if the piece of data is a byte). If the data is not a byte, this field MUST be NULL.

vast
: The vast value shows the value of the data (if the piece of data is of type int or vast). If the data is not an int or vast, this field MUST be NULL.

uvast
: The uvast value shows the value of the data (if the piece of data is of type uint or vast). If the data is not an uint or uvast, this field MUST be NULL.

real
: The real value shows the value of the data (if the piece of data is of type real32 or real64). If the data is not a real32 or real64, this field MUST be NULL.

Str
: The str field shows the value of the data (if the piece of data is of type string). If the data is not a string, this field MUST be NULL.

Bool
: The bool field shows the value of the data (if the piece of data is of type boolean). If the data is not a boolean, this field MUST be NULL.

ts

The ts field shows the value of the data (if the piece of data is of type timestamp). If the data is not a timestamp, this field MUST be NULL.

6.1.2. Data Collection - data_set

The data_collection_set table holds information for a particular collection of data entries (from data_value). The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
data_id	int(10) unsigned	NO	PRI		auto_increment
comment	varchar(255)	NO			

data_set

data_id
: The Data Collection identifier. This is a primary key.

comment
: This is a description of the data set.

6.2. ARI Collections

A ARI Collection is an ordered set of ARI values, similar to a Data Collection, which is an ordered set of Data values. One table is used to represent the ARI Collection, and another table is used to capture the ordered ARIs in the collection.

6.2.1. ARI Collection Entry - ac

The ac table holds ARI collection entries, one per row. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
ac_id	int(10) unsigned	NO	MUL		
ac_order	int(10) unsigned	NO		0	
ari_id	int(10) unsigned	NO	MUL		

ac

ac_id

The ARI Collection to which this entry belongs. This is a foreign key that corresponds with the ac_id of the ac_set table.

ari_id
: The identifier for this ARI. This is a foreign key that corresponds to the ari_definition table.

ac_order
: The 0-based order of this entry in the encapsulating collection.

6.2.2. ARI Collection - ac_set

The ac_set table holds information for a particular collection of ARIs (from the ac table). The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
ac_id	int(10) unsigned	NO	PRI		auto_increment
comment	varchar(255)	NO			

ac_set

ac_id
: The ARI Collection identifier.

7. Operator Support

7.1. In Type Set - in_type_set

The in_type_set table holds information for a particular collection of operator in types (from the in_types table). The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
in_type_id	int(10) unsigned	NO	PRI		auto_increment
comment	varchar(255)	NO			

in_type_set

in_type_id

The in_type set identifier.

comment
The description of the in_type set.

7.2. In Type - in_type

The in_type table holds information about operator in types. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
in_type_id	int(10) unsigned	NO	PRI		auto_increment
name	varchar(255)	NO			
order	int(10) unsigned	NO			
type id	int(10) unsigned	NO	MUL		

in_type_set

in_type_id
The in_type set identifier. This is a foreign key that corresponds to the in_type_id of the in_type_set table.

name
The name of the operator in type.

order
The order of the operator in type in the set.

type id
This is the type of the operator in_type. This is a foreign key that corresponds to the enumeration in the data_types table.

8. Outgoing Message Support

Outgoing messages are those that are written into the database by an application and read by an AMP Manager, formatted, and sent to one or more AMP Agents.

The database represents outgoing messages in two tables. One table holds information for the entire outgoing message group, and another table captures each individual outgoing message. An individual outgoing message is simply a ARI collection of the ARIs to be run on the Agent.

8.1. Outgoing Messages - outgoing_message

The outgoing_message table captures a single ARI Collection holding the set of Control ARIs to be sent to an Agent as part of a Message Group. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
message_id	int(10) unsigned	NO	PRI		auto_increment
group_id	int(10) unsigned	NO	MUL		
start_ts	datetime	NO		0	
ac_id	int(10) unsigned	NO	MUL		

outgoing_message

message_id
: The primary key for this table.

group_id
: The outgoing message group to which this outgoing message belongs. This is a foreign key that corresponds to the id in the outgoing_message_group table.

start_ts
: The time at which the controls in the ARI Collection should be run. This may be an absolute or relative time, as defined in [I-D.birrane-dtn-amp].

ac_id
: The ARI Collection comprising this outgoing message. This is a foreign key that corresponds to the ac_id of the ac_set table.

8.2. Outgoing Message Groups - outgoing_message_group

The outgoing_message_group table captures an outgoing message group, which is one or more outgoing messages. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
group_id	int(10) unsigned	NO	PRI		auto_increment
created_ts	datetime	YES		NULL	
modified_ts	datetime	YES		NULL	
state	tinyint(4)	NO	MUL		
agent_id	int(10) unsigned	NO	MUL		

outgoing_message_group

group_id
: The primary key for this table.

created_ts
: The time at which the outgoing message group was created.

modified_ts
: The last time at which the message group was modified.

state
: The current state of the outgoing message group. This state provides a contention-avoidance mechanism between an application and an AMP Manager. This is a foreign key that corresponds to the state_id of the outgoing_state table.

agent_id
: The identifier of the Agent receiving this message group. Currently, a message group is only sent to one Agent. Sending to multiple Agents is accomplished with multiple entries in this table. This is a foreign key that corresponds to the registry_id in the registered_agents table.

9. Incoming Message Support

Incoming messages are those that are written into the database by an AMP Manager and read by an application wishing to understand the status of an AMP Agent.

The database represents incoming messages in two tables. One table holds information for the entire incoming message group, and another table captures each individual incoming message.

9.1. Incoming Messages - incoming_messages

The incoming_messages table captures information returned from an AMP Agent. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
message_id	int(10) unsigned	NO	PRI	NULL	auto_increment
group_id	int(10) unsigned	NO	MUL	NULL	
ac_id	int(10) unsigned	YES	MUL	NULL	

incoming_messages

message_id
: The primary key for this table.

group_id
: The incoming message group to which this incoming message belongs. This is a foreign key that corresponds with the group_id in the incoming_message_group table.

ac_id
: This is a foreign key that corresponds to the id in the ac_set table.

9.2. Incoming Message Groups - incoming_message_group

The incoming_message_group table captures an incoming message group, which is one or more incoming messages. The format of the table is defined as follows.

Field	Type	Null	Key	Default	Extra
group_id	int(10) unsigned	NO	PRI		auto_incr
recieved_ts	datetime	YES		NULL	
generated_ts	datetime	YES		NULL	
state	tinyint(3) unsigned	NO	MUL		
agent_id	int(10) unsigned	NO	MUL		

incoming_message_group

group_id
: The primary key for this table.

recieved_ts
: The time at which the incoming message group was received by the AMP Manager.

generated_ts
: The time at which the incoming message group was generated by the sending AMP Agent.

state
: The current state of the incoming message group. This state provides a contention-avoidance mechanism between an application and an AMP Manager. This is a foreign key that corresponds to the data_id in the data_types table.

agent_id
: The identifier of the Agent sending this incoming message group. This is a foreign key that corresponds to the registry_id in the registered_agents table.

10. Compound Data Types

10.1. TNVC Support

10.1.1. Type Name Value Collection - tnvc

The tnvc table is a group of tnv collections.

Field	Type	Null	Key	Default	Extra
tnvc_id	int(10) unsigned	NO	PRI		auto_incr
order	int(10) unsigned	NO	UNI		
name	int(10) unsigned	YES			
value	int(10) unsigned	NO	MUL		

tnvc

tnvc_id
: The identifier of the TNV collection. This is a foreign key that corresponds to the tnvc_id in the tnvc_set table.

order
: The order of the tnvc in the set.

name
: The name of the tnvc.

value
: The value of the tnvc. This is a foreign key that corresponds to the data_id in the data_set table.

10.1.2. Type Name Value Collection Set - tnvc_set

The tnvc_set table provides the information of a TNV (Type-Name-Value) collection that describes data values in the AMM.

Field	Type	Null	Key	Default	Extra
tnvc_id	int(10) unsigned	NO	PRI	NULL	auto_incr
comment	varchar(255)	NO			

tnv_collection

tnvc_id
: The identifier of the TNV collection. This is a primary key.

comment
: This is the description of the tnvc set.

10.2. Expression Support

10.2.1. Expression Set - expression_set

The expression_set table shows the id of the expression as well as whether it is true or false. This table is a collection of expressions.

Field	Type	Null	Key	Default	Extra
expr_id	int(10) unsigned	NO	PRI	NULL	auto_incr
comment	varchar(255)	NO			

expression_set

expr_id
: The identifier of the expression. This is a primary key.

comment
: This is the description of the expression.

10.2.2. Individual Expression - expression

An expression is an AC in which a series of items are ordered so as to produce a valid post-fix mathematical expression. This table contains all of the information that are included in each expression.

Field	Type	Null	Key	Default	Extra
expr_id	int(10) unsigned	NO	PRI	NULL	auto_incr
order_num	int(10) unsigned	NO	UNI	NULL	
ari_id	int(10) unsigned	NO	MUL	NULL	

expression

expr_id
: The identifier of the expression. This is a foreign key that corresponds to the expr_id in the expression_set table.

order_num
: The 0-based order of this entry in the encapsulating collection.

ari_id
: This is the ari id. This is a foreign key that corresponds to the ari_id in the ari_definition table.

11. IANA Considerations

At this time, this schema definition has no fields registered by IANA.

12. Security Considerations

Security considerations are outside of the scope of this document.

13. References

13.1. Informative References

[I-D.birrane-dtn-ama]
Birrane, E., "Asynchronous Management Architecture", draft-birrane-dtn-ama-07 (work in progress), June 2018.

13.2. Normative References

[I-D.birrane-dtn-adm]
Birrane, E., DiPietro, E., and D. Linko, "AMA Application Data Model", draft-birrane-dtn-adm-02 (work in progress), June 2018.

[I-D.birrane-dtn-amp]
Birrane, E., "Asynchronous Management Protocol", draft-birrane-dtn-amp-04 (work in progress), June 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Appendix A. Acknowledgements

The following participants contributed technical material, use cases, and useful thoughts on the overall approach to this database specification: Leor Bleir of the NASA Goddard Space Flight Center, Michael Deschu and Shane Knudsen of Hammers Company, Inc. on behalf of the NASA Goddard Space Flight Center, and Paul Swencon of ASRC Space And Defense on behalf of the NASA Goddard Space Flight Center.

Authors' Addresses

Edward J. Birrane
Johns Hopkins Applied Physics Laboratory

Email: Edward.Birrane@jhuapl.edu

Evana DiPietro
Johns Hopkins Applied Physics Laboratory

Email: Evana.DiPietro@jhuapl.edu

David Linko
Johns Hopkins Applied Physics Laboratory

Email: David.Linko@jhuapl.edu

Mark A. Sinkiat
ASRC Space And Defense, NASA GSFC

Email: mark.a.sinkiat@nasa.gov