Workgroup: Delay-Tolerant Networking
Internet-Draft: draft-birrane-dtn-ari-00
Published: 17 December 2021
Intended Status: Standards Track
Expires: 20 June 2022
Authors: E.J. Birrane
JHU/APL
E.A. Annis
Johns Hopkins Applied Physics Laboratory

AMM Resource Identifier

Abstract

This document defines the structure, format, and features of the naming scheme for the objects defined in the Delay-Tolerant Networking Autonomous Management Model (AMM), in support of challenged network management solutions described in the Delay-Tolerant Networking Autonomous Management Architecture (AMA).

This document defines a new AMM Resource Identifier (ARI), based on the structure of a common URI, meeting the needs for a concise, typed, parameterized, and hierarchically organized set of data elements.

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 20 June 2022.

Copyright Notice

Table of Contents

1. Introduction

The unique limitations of Delay-Tolerant Networking (DTN) transport capabilities [RFC4838] necessitate increased reliance on individual node behavior. These limitations are considered part of the expected operational environment of the system and, thus, contemporaneous end-to-end data exchange cannot be considered a requirement for successful communication.

The primary DTN transport mechanism, Bundle Protocol version 7, (BPv7) [I-D.ietf-dtn-bpbis], standardizes a store-and-forward behavior required to communicate effectively between endpoints that may never co-exist in a single network partition. BPv7 might be deployed in static environments, but the design and operation of BPv7 cannot presume that to be the case.

Similarly, the management of any BPv7 protocol agent (BPA) (or any software reliant upon DTN for its communication) cannot presume to operate in a resourced, connected network. Just as DTN transport must be delay-tolerant, DTN network management must also be delay-tolerant.

The DTN Autonomous Management Architecture (DTN AMA) [I-D.ietf-dtn-ama] outlines an architecture that achieves this result through the self-management of a DTN node as configured by one or more remote managers in an asynchronous and open-loop system. An important part of this architecture is the definition of a conceptual data schema for defining resources configured by remote managers and implemented by the local autonomy of a DTN node.

The DTN Asynchronous Management Model (DTN AMM) [I-D.birrane-dtn-adm] defines a logical schema that can be used to represent data types and structures, autonomous controls, and other kinds of information expected to be required for the local management of a DTN node. The DTN AMM further describes a physical data model, called the Application Data Model, that can be defined in the context of applications to create resources in accordance with the DTN AMM logical schema. These named resources can be predefined in moderated publications or custom-defined as part of the operational management of a network or a node.

Every AMM resource must be uniquely identifiable. To accomplish this, an expressive naming scheme is required. The AMM Resource Identifier (ARI) provides this naming scheme. This document defines an ARI, based on the structure of a URI, meeting the needs for a concise, typed, parameterized, and hierarchically organized naming convention.

1.1. Scope

The ARI scheme is based on the structure of a URI [RFC3986] in accordance with the practices outlined in [RFC8820].

ARIs are designed to support the identification requirements of the DTN AMM logical schema. As such, this specification will discuss these requirements to the extent necessary to explain the structure and use of the ARI syntax.

This specification does not constrain the syntax or structure of any existing URI (or part thereof). As such, the ARI scheme does not impede the ownership of any other URI definition and is therefore clear of the concerns presented in [RFC7320].

This specification does not discuss the manner in which ARIs might be generated, populated, and used by applications. The operational utility and configuration of ARIs in a system are described in other documents associated with DTN management, to include the AMA and AMM specifications.

This specification does not describe the way in which path prefixes associated with an ARI are standardized, moderated, or otherwise populated. Path suffixes may be specified where they do not lead to collision or ambiguity.

This specification does not describe the mechanisms for generating either standardized or custom ARIs in the context of any given application, protocol, or network.

This specification does not describe the ways in which an ARI could be encoded into other formats, to include compressed binary formats. However, the design of the ARI syntax discusses compressibility to the extent that the design impacts the ability to create such encodings.

2. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

*DTN Autonomous Management Model (AMM) - data types and data structures needed to manage applications in challenged networks.

* AMM Resource Identifier (ARI) - A unique identifier for any AMM object, syntactically conformant to the Uniform Resource Identifier (URI) syntax documented in [RFC3986] and using the scheme name "ari".

* AMM Namespace - A moderated, hierarchical taxonomy of namespaces that describe a set of AMM scopes. Specifically, an individual AMM namespace is a specific sequence of AMM namespaces, from most general to most specific, that uniquely and unambiguously identify the namespace of a particular AMM.

* Operational Data Model (ODM) - The operational configuration of an Agent. This includes the union of all ADM information supported by the Agent as well as all operational, dynamic configuration applied to the Agent by Managers in the network.

4. ARI Scheme Utility

AMM resources are referenced in the context of autonomous applications on an agent. The naming scheme of these resources must support certain features to inform AMA processing in accordance with the AMM logical schema.

This section defines the set of unique characteristics of the ARI scheme, the combination of which provides a unique utility for naming. While certain other naming schemes might incorporate certain elements, there are no such schemes that both support needed features and exclude prohibited features.

4.1. Resource Parameterization

The AMM schema allows for the parameterization of resources to both reduce the overall data volume communicated between DTN nodes and to remove the need for any round-trip data negotiation.

Parameterization reduces the communicated data volume when parameters are used as filter criteria. By associating a parameter with a data source, data characteristic, or other differentiating attribute, DTN nodes can locally process parameters to construct the minimal set of information to either process for local autonomy or report to remote managers in the network.

Parameterization eliminates the need for round-trip negotiation to identify where information is located or how it should be accessed. When parameters define the ability to perform an associative lookup of a value, the index or location of the data at a particular DTN node can be resolved locally as part of the local autonomy of the node and not communicated back to a remote manager.

4.2. Compressible Structure

The ability to encode information in very concise formats enables DTN communications in a variety of ways. Reduced message sizes increase the likelihood of message delivery, require fewer processing resources to secure, store, and forward, and require less resources to transmit.

While the encoding of an ARI is outside of the scope of this document, the structure of portions of the ARI syntax lend themselves to better compressibility. For example, DTN AMM encodings support the ability to identify resources in as few as 3 bytes by exploiting the compressible structure of the ARI.

The ARI syntax supports three design elements to aid in the creation of more concise encodings: relative paths, path suffixing, and patterning.

4.2.1. Relative Paths

Hierarchical structures are well known to support compressible encodings by strategically enumerating well-known branching points in a hierarchy. For this reason, the ARI syntax uses the URI path to implement a naming hierarchy.

Supporting relative paths allow for the ARI namespace to be shortened relative to a well-known prefix. By eliminating the need to repeat common path prefixes in ARIs (in any encoding) the size of any given ARI can be reduced.

This relative prefix might be relative to an existing location, such as the familiar "../item" or relative to a defined nickname for a particular path prefix, such as "{root}/item".

4.2.2. Path Suffixing

Path suffixing refers to specifying how information close to the "leaf" node of a hierarchy is structured. By codifying this structure into the ARI syntax, elements of the AMM logical scheme can be enumerated and compressed in the same way for any physical data model instantiation of an ARI.

4.2.3. Patterning

Patterning in this context refers to the structuring of ARI information to allow for meaning data selection as a function of wildcards, regular expressions, and other expressions of a pattern.

Patterns allow for both better compression and fewer ARI representations by allowing a single ARI pattern to stand-in for a variety of actual ARIs.

This benefit is best achieved when the structure of the ARI is both expressive enough to include information that is useful to pattern match, and regular enough to understand how to create these patterns.

5. ARI Component Definitions

This section describes the components of the ARI scheme to inform the discussion of the ARI syntax in Section Section 6. These components include ARI Namespaces, Object Names, Parameters, and Special Representations.

5.1. Namespaces

AMM resources exist within namespaces to eliminate the chance of a conflicting resource name, aid in the application of patterns, and improve the compressibility of the ARI. Namespaces MUST NOT be used as a security mechanism. An Agent or Manager MUST NOT infer security information or access control based solely on namespace information in an ARI.

The AMM defines two types of namespaces whose representation within an ARI is slight different: Regular Namespaces and Anonymous Namespaces.

5.1.1. Anonymous Namespaces

The ARI syntax supports the definition of AMM resources absent a containing namespace. In this sense, the resource is considered "anonymous" in that it is not placeable in a particular hierarchy and, thus, not able to be located based on relative paths, patterns over the namespace hierarchy, or other characteristic based on the namespace.

Anonymous namespaces are most effectively used for the representation of literal values and constants that have utility and definition that is not otherwise associated with a single namespace.

For example, the representation of the strongly typed integer value 4 could be representing using the anonymous namespace as: ari:uint(4)

5.1.2. Regular Namespaces

A regular namespace is simply any namespace other than the empty namespace reserved for anonymous ARIs.

Namespaces are hierarchical, which allows the grouping of resources that share common attributes - for example, resources associated with related protocols may have protocol-specific namespaces that are grouped under a single encompassing namespace. Namespaces that are closer to a "root node" in a hierarchy have broader scope than namespaces closer to "leaf nodes" in the same hierarchy.

In a hierarchical model of namespaces, a particular namespace is identified as the path to that namespace through the hierarchy. The ARI encodes this path with the URI path attribute.

The ARI scheme defines two types of namespaces: moderated and informal.

5.1.2.1. Moderated Namespaces

Moderated namespaces identify AMM resources that have been defined in a normative, moderated way by some standards organization. These resources are immutable and can be relied on the be define and used the same way across multiple networks and multiple implementations.

Because the source of the resource definition in a moderated namespace represents an authoritative reference to this object, moderated namespaces always include an authority segment.

5.1.2.2. Informal Namespaces

Informal namespaces represent resources that are only defined in the context of a specific network, mission, or application or those resources that might only be defined for a certain period of time.

Unlike moderated namespaces, informal namespaces have no defined authority associated with them. The path representing these namespaces may be any valid path.

The general form of an informal namespace is given as: <ISSUER>/<TAG>.

An Issuer is any string that identifies the organization that is defining an AMM object. This value may come from a global registry of organizations, an issuing node address, a signed known value, or some other network-unique marking. Issuers MUST NOT conflict with known moderated namespaces, and AMA Agents and Managers should not process Issuers that conflict with existing moderated namespaces.

A Tag is any (optional) string used to disambiguate AMM Objects for an Issuer. The contents of the tag are left to the discretion of the Issuer. Examples of potential tag values include an issuer-known version number or a (signed) hashing of the data item associated with the reference identifier.

5.2. Object

An object is any one of a number of data elements defined for the management of a given application or protocol that conforms to the AMM logical schema.

Objects are identified in the ARI scheme by the catenation of their AMM logical schema type and a string name. Additionally, objects may be further differentiated by any parameters defined for that object.

5.3. Parameters

The AMM logical schema allows many object types to be parameterized when defined in the context of an application or a protocol.

If two instances of an AMM resource have the same namespace and same object type and object name but have different parameter values, then those instances are unique and the ARIs for those instances MUST also be unique. Therefore, parameters are considered part of the ARI syntax.

The AMM logical schema defines two types of parameters: Formal and Actual. The terms formal parameter and actual parameter follow common computer programming vernacular for discussing function declarations and function calls, respectively.

5.3.1. Formal Parameters

Formal parameters define the type, name, and order of the information that customizes an ARI. They represent the unchanging "definition" of the parameterized object.

Formal parameters MUST include type and name information and MAY include an optional default value. If specified, a default value will be used whenever a set of actual parameters fails to provide a value for this formal parameter.

5.3.2. Actual Parameters

Actual parameters represent the data values used to distinguish different instances of a parameterized object.

An actual parameter MUST specify a value and MAY specify a type. If a type is provided it MUST match the type provided by the formal parameter. An actual parameter MUST NOT include NAME information.

Including type information in an actual parameters allows for explicit type checking of a value, which might otherwise be implicitely cast.

There are two ways in which the value of an actual parameter can be specified: parameter-by-value and parameter-by-name.

Parameter-By-Value

This method involves directly supplying the value as part of the actual parameter. It is the default method for supplying values.

Parameter-By-Name

This method involves specifying the name of some other parameter and using that other parameter's value for the value of this parameter. This method is useful when a parameterized ARI contains another parameterized ARI. The contained object's actual parameter can be given as the name of the containing ARI's parameter. In that way, a containing ARI's parameters can be "flowed down" to all of the objects it contains.

NOTE: In cases where a formal parameter contains a default value, the associated actual parameter may be omitted. Default values in formal parameters (and, thus, optional actual parameters) are encouraged as they reduce the size of data items communicated amongst Managers and Agents in a network.

5.4. Special Case: Literal Values

A literal value is one whose value and name are equivalent. The name is, literally, the value. For example, the literal "4" serves as both a name and a value. When literal values are represented, the object name MUST be omitted and the literal value substituted as a parameterization of the object.

Further, because the value of a Literal serves as its name, there is no need for regular namespaces. All literals exist in the anonymous namespace.

6. ARI Scheme Syntax

There are three components to the ARI: namespaces, objects, and parameters. This section defines the syntactic representation of each of these components, and discusses special cases.

The scheme name of the ARI is "ari". The scheme-specific part of the "ari" scheme follows the format: ari:/<Namespace>/<Object><(Parameters)> The string representation of each component is given as follows.

Namespaces

Namespaces are represented as "/" separated lists, with individual namespace types represented as follows:

* Moderated namespaces are listed using a URI authority representing the normative moderator for the resource followed by a URI path relative to that moderator.

  For example: "ari://sdo/ietf/dtn/adm/bp/".

* Anonymous namespaces are empty and are represented as the lack of a starting / or //.

  For example: "ari:type.name(parm)".

* Informal namespaces are URI paths without a URI authority present.

  For example: "ari:/myproject/dtn/bp/dynamic".

Objects

The object name is represented as the two-tuple of the object type and the object name, joined with the '.' character.

For example: "uint.num_bundles".

Parameters

If present, parameters are represented as a comma-separated string enclosed in parenthesis. Different types of parameters are represented as follows.

* Formal parameters follow the pattern <type> <name> and if there is a default value, it is represented by the substring "= <value>".

* Actual Parameters-By-Value are represented as the string encoding of their value.

* Actual Parameters-By-Name are represented as the name of the parameter enclosed in angle brackets.

Note: If an actual parameter is missing for a formal parameter that has a default value, then the ARI MUST have a blank space where the actual parameter would have been. This missing parameter will also have a comma, separating it from other actual parameters in the ARI string.

6.1. Literal String Encoding

The string representation of a Literal ARI is much simpler and consists of simply the data type of the Literal followed by the value, as follows:

"ari:/type(value)"

6.2. Delimiting Characters

For the scheme specific parts, there is no authority to be defined for the ARI URI. The scheme is separated from the path using a ":" and the path components are separated and terminated using a "/". The path is comprised of the namespaces which hierarchally organize the AMM objects. The object is defined by both its AMM object type (such as EDD, VAR, RPTT, etc.) and the object name, each separated by a ".". The object parameters are separated using reserved characters "{", "}", "[", "]", "(", ")" described below. Finally the "#" sign is used at the end of the ARI only in cases where custom issuer specific objects are defined. Example = ari:/top-a/mid-c/low-b/edd.dtnObject([int dtnObjParam1], [str dtnObjParam2])#custom-issuer-1

6.2.1. Wildcards

TBD

7. Encoding Considerations

ARIs might be represented in a variety of different formats, to include human-readable strings, structured language representations (such as XML or JSON), and binary encodings (such as CBOR). An ARI scheme must support the mechanical translation amongst this diversity of representations.

An ARI scheme should represent, and differentiate, different kinds of information using a standard format. Such standard formats should rely on delimiters and other structural components and not informal naming conventions.

8. Scheme Registration Considerations

This section contains fields required for the URI scheme registration, following the guidelines in [RFC7595]

TODO: Define characters used for globs, wildcards, expression matching, etc.

9. Interoperability Considerations

DTN challenged networks might interface with better resourced networks that are managed using non-DTN management protocols. When this occurs, the federated network architecture might need to define management gateways that translate between DTN and non-DTN management approaches.

NOTE: It is also possible for DTN management be used end-to-end because this approach can also operate in less challenged networks. The opposite is not true; non-DTN management approaches should not be assumed to work in DTN challenged networks.

Where possible, ARIs should be translatable to other, non-DTN management naming schemes. This translation might not be 1-1, as the features of the AMM may differ from features in other management naming schemes. Therefore, it is unlikely that a single naming scheme can be used for both DTN and non-DTN management.

10. Security Considerations

Policy decisions as to whether anonymous namespaces are allowed in the system should be determined before network deployment. The use of an anonymous namespace greatly increases the chances of naming collisions.

Because informal namespaces are defined by any entity, no security or permission meaning can be inferred simply from the expression of namespace.

11. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of schema and namespaces related to the AMM Resource Identifier (ARI), in accordance with BCP 26 [RFC1155].

11.1. ARI Scheme

This document defines a new URI scheme, "ari", as defined in Section 6. This scheme to be registered by IANA here: https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml

12. References

12.1. Normative References

[I-D.ietf-dtn-bpbis] Burleigh, S., Fall, K., and E. Birrane, "Bundle Protocol Version 7", Work in Progress, Internet-Draft, draft-ietf-dtn-bpbis-31, 25 January 2021, <https://

www.ietf.org/internet-drafts/draft-ietf-dtn-bpbis-31.txt>.

[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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/info/rfc7595>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[I-D.birrane-dtn-adm] Birrane, E., DiPietro, E., and D. Linko, "AMA Application Data Model", Work in Progress, Internet-Draft, draft-birrane-dtn-adm-03, 2 July 2018, <http://www.ietf.org/internet-drafts/draft-birrane-dtn-adm-03.txt>.

[I-D.ietf-dtn-ama] Birrane, E. J., Annis, J. E., and S. Heiner, "Asynchronous Management Architecture", Work in Progress, Internet-Draft, draft-ietf-dtn-ama-03, 25 October 2021, <https://www.ietf.org/archive/id/draft-ietf-dtn-ama-03.txt>.

[RFC1155] Rose, M. and K. McCloghrie, "Structure and identification of management information for TCP/IP-based internets", STD 16, RFC 1155, DOI 10.17487/RFC1155, May 1990, <https://www.rfc-editor.org/info/rfc1155>.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <https://www.rfc-editor.org/info/rfc4838>.

[RFC7320] Nottingham, M., "URI Design and Ownership", RFC 7320, DOI 10.17487/RFC7320, July 2014, <https://www.rfc-editor.org/info/rfc7320>.

[RFC8820]
Nottingham, M., "URI Design and Ownership", BCP 190, RFC 8820, DOI 10.17487/RFC8820, June 2020, <https://www.rfc-editor.org/info/rfc8820>.

Appendix A. Examples

A.1. Namespace Examples

For example, consider the namespaces in Figure 1.

ARI Namespace Hierarchy

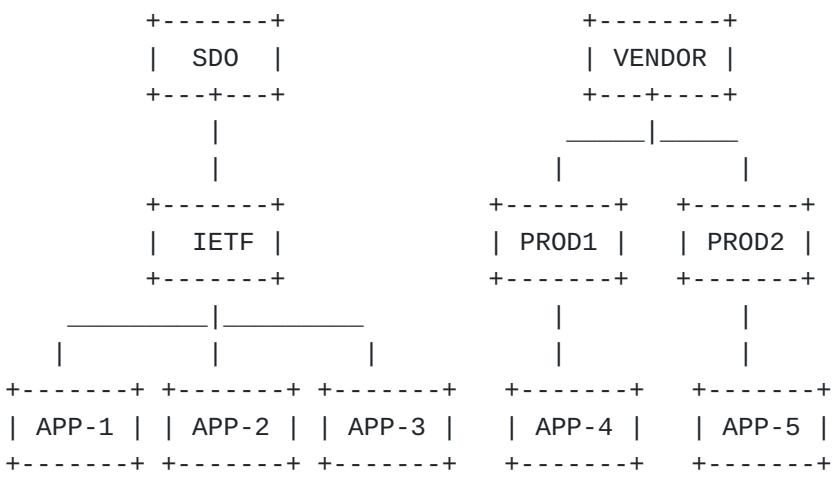

Figure 1

Given this hierarchy, the following are all valid namespace representations.

ari://sdo/ietf/

ari://sdo/ietf/app-1/

ari://sdo/ietf/app-3/

ari:/vendor/

ari:/vendor/prod1/

ari:/vendor/prod2/app-5

ari:/

A.2. Object Examples

The ARIs for the following sample AMM objects are encoded in Table 1. Note that these examples are for the identifiers of AMM objects, not their entire definition.

*The number of bytes received by an Agent, defined in the N1/N2 namespace and called num_bytes.

*The number of bytes received through an interface, called num_bytes_if, which takes a single string parameter named "if_name" with a default value oth "eth0".

*An anonymous, operator-defined object named "obj1" which takes two unsigned integer parameters, n1 and n2, with default values of 3 and 4, respectively.

*The typed, Literal value of 4.

ARI String	Description
"ari:/N1/N2/num_bytes"	Unparameterized num_bytes object in the N1/N2 informal namespace.
"num_bytes"	Shortform encoding where the N1/N2 namespace can be assumed.
"num_bytes_if(String if_name)"	Formal parameter definition of num_bytes object that accepts a string interface name.
"num_bytes_if(String if_name=eth0)"	Formal parameter definition of num_bytes object that accepts a string interface name with a default value.
"num_bytes_if()"	Actual parameter using the default value of eth0.
"num_bytes_if(eth0)"	Actual parameter of eth0.
"ari:/obj1(Int n1 = 0, Int n2 = 3)"	Formal parameter of object obj1 in anonymous namespace taking 2 default parameters.
"ari:/obj1(,)"	Actual parameter using the default values of 0 for n1 and 3 for n2.
"ari:/obj1(, 4)"	Actual parameter using the default value of 0 for n1.
"ari:/obj1(4,)"	Actual parameter using the default value of 3 for n2.
"ari:/obj1(4,4)"	Actual parameters provided for all obj1 parameters.
"ari:/obj1(<input>,4)"	Actual parameters provided for all obj1 parameters, with the value of the first parameter taken from some other parameter named "input".
"ari:uint(4)"	The Literal value 4 interpreted as a 32-bit unsigned integer.

Table 1

Authors' Addresses

Edward J. Birrane, III
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Rd.
Laurel, MD 20723
United States of America

Phone: +1 443 778 7423
Email: Edward.Birrane@jhuapl.edu

Emery Annis
Johns Hopkins Applied Physics Laboratory

Email: Emery.Annis@jhuapl.edu