Diameter Specification Reccomendations

IETF 99
July 2017
(draft-bertz-dime-diamimpr-00)

Motivation

- As an Operator, Sprint was observing successful interop of Diameter but long lead times for development
- Sprint's C3PO project (a 4G open source EPC developed jointly with Intel) uses freediameter
  - Also noticed longer than expected implementation lead times
  - Errors were minor but enough to be a drag on the schedule
  - We decided to do a study to determine how efficient we can be
- Hypothesis – We can go from spec to code (library + structs / objects + json marshaling +...) < 1 hour

Findings

- We were so wrong...
- Too many table formats
  - It cannot be argued this is a Change over time, some specs started / became RFCs at similar times but have different Table formats
- Imports of AVPs not treated consistently
- Outright errors slow us down
  - Some AVPs NEVER given a Code
  - Some AVPs referenced and missing (NEVER defined)
- Some data you just can't get from any spec programmatically
  - App ID + Messages + Return Codes + App Name
  - Especially hard when a spec has multiple applications

Methodology

- Save Document in text format
- Put Defined AVPS and Imported/Referenced/Re-used AVPs in different files
- Add Enums ONLY if parser cannot find them
- Put app info in a special file to map commands, return codes, app id, app name, etc.
- Run python code to create 'dia' files used in diafuzzer (https://github.com/Orange-OpenSource/diafuzzer)
  - Enhance as required to support use cases
- Load the dia files into memory (which forces many semantic checks)

Unexpected Use Cases

- Grouped AVPs/Commands that further refine the AVPs/Commands defined in other applications
- Easy to detect
  - Except when in the same document (that was no fun)
- Some inconsistencies - in some specs the Grouped AVP was Imported and others it was defined when in fact it is both
- There is no concept of imported commands...

We are not immune

- RFC 4006 bis had a few minor issues (to be closed as part of WGLC)

Enumerations

- Don't use them. Just don't...
- Use Cases
  - Enums referencing other Enums
    - Some reuse Enums
    - Others restrict Enums – can you do that?
    - Others add new values
  - Enums referencing web pages (URIs)
  - Enums reference registries (which is expected)
- No format and very inconsistent
  - So inconsistent we often just appended enums to generated files and manually processed them

Recommendations

- Overall:
  - The name of all AVPs, Commands and Grouped AVPs appear consistently throughout the docume
  - The letter case MUST be consistent for all names
  - No spaces should appear in the names.
  - Use of underscores is discouraged except for line continuations in tables and enumeration labels.

Defined AVP Recommendations

- Tables MUST include the following columns:
  - Attribute Name
  - AVP Code
  - Section Defined
  - Data Type
  - AVP Flag Rules for MUST and MUST NOT
  - Other minor recommendations in spec

Defined AVPs Recommended Formats

Attribute Name	AVP Code	Section Defined	Data Type	MUST	MUST NOT
AVP-Name	85	9.8.2	Unsigned32	M	V

Example Two

Attribute Name	AVP Code	Section Defined	Data Type	MUST	MUST NOT
AVP-Name	85	9.8.2	Unsigned32	M	V

Imported AVP Recommendations

- Imported or Re-used AVPs MUST be included in the specification. A table MUST be present if AVPs are re-used/imported.
  - The table MUST include the AVP and Source document columns.
  - The table MAY include a Comment column.
  - An M-bit column MAY be present as required.
    - (Should this be MAY or MUST?)
  - The table MUST be pipe delimited when in text format.

Grouped AVP / Command Refinement

- Preconditions
  - Original structure MUST have *[AVP] present to be Refinable (see Open Questions later in presentation about this)
  - Any refinement of an AVP present the new structure MUST conform to the occurrences range
    - If X*Y[Foo] was present in original and A*B [Foo] present in refinement then X <= A <= B <= Y.
  - AVPs retained without further restriction of the number of occurrences MUST be kept in the Refining AVP's definition otherwise they are assumed to be dropped from the new AVPdefinition.
- Refinement is defined by 'Refines [App Id]'. When App Id is not present, it is assumed to be the original spec.
  - A Refined AVP / Command MUST NOT appear in anything but an Import Table EXCEPT for the specification that originally defined the AVP / Command.
    - Allows for easy detection of defining spec but not helpful when a define/refine is in the same document.<< Open question

Example Refinement

From TS 29.336

User-Identifier ::= <AVP-Header: 3102, 10415>
[User-Name]
[MSISDN]
[External-Identifier]
[LMSI]
*[AVP]

From TS 29.128

User-Identifier ::= <AVP-Header: 3102, 10415, **Refines**>
[User-Name]
*[AVP]

Command Recommendations

- inconsistent values between the name, three letter acronym defined in the table and the actual name used in the command definition.

Enumeration Recommendations

- Enumeration Value Names MUST adhere to alphanumeric and underscore characters.
- Enumeration Value Names MUST not begin with an underscore.
- When being defined the format MUST include the label and the value assigned with the label enclosed in parenthesis on a single line.
  - Example w/o parenthesis
    - Speed_10 10
    - Speed_1010 (Error)
  - Example with parenthesis
    - Speed_10 (10)
    - Speed_10(10)
- Coverage of Use Cases were NOT specified in the document

GAPs for Automated Validation

Many pieces of information cannot be programmatically validated.

GAP 1: The application identifier and name of an application.

GAP 2: The application and vendor identifiers associated with a defined AVP table.

GAP 3: The application and vendor identifiers associated with Commands.

GAP 4: Reused and newly defined result codes for an application.

GAP 5: Easily parsed enumerations that cover all use cases.

Example to Close Gaps

- Just a suggestion here

1: AppFoo ::= <Diameter Application: 10415 101010>
2: Command1-Name-Request C1R
3: Command1-Name-Answer C1A
4:
5: Result-Codes ::= <Diameter Result-Codes: 101010>
6: NEW_RESULT (4999)
7: IMPORTED_RESULT IMPORT (4010)

GAP 1 is closed in line 1. GAP 3 is closed in lines 1 through 3 while GAP 4 is closed by lines 5 through 7.

GAP 2 can be closed by using a common discernable Table Name format, e.g. AppFoo defined AVPs. In this case the Application Name can be looked up and associated to the defined AVP table.

Enumeration Example Format

Gap 5 can be partially closed by following a pattern similar to Result-Codes but this does not resolve all uses cases.

```
Result-Codes ::= <Diameter Enumeration: 123, 45678>
   Label_1 (0)
   LABEL_Two (2)
```

Does not resolve Use Cases.

Open Questions

- Open question, can a Grouped AVP/Command have a range limited [AVP] member, e.g. *5[AVP]?
- Do we go back and file Errata on all items noted in the Survey?
  - Will we take up fixing these?
- Do we want to mix 'pseudo enumerations' (defined as Unsigned32 but have a spec for mapping labels to values and DOES NOT require a registry)
- When a define/refine is in the same document we know it should be in a Defined Grouped AVP (if it is a Grouped AVP) but should we make other considerations?

Summary

- Diameter over the wire is fine
- The specs are doing well (our evidence is that over the wire interoperability works)
- We can do better!
- Our current formats cannot cover all use cases
- Automated Validation is possible
- Good specs take less than 20 minutes to go from spec to code generation

Next Steps

- WG Adopt this document as either

  A> a requirements document and make changes in another document (my recommendation)

  B> a solutions document (much more work required)
- File Errata based upon results but don't work on fixes unless the specs are actually used
- Figure out what to do for Enums (add as WG Item)
- Work on an automated verification system as part of the I-D nits process
  - Should it be gating all of the time or only for WGLC?

Diameter Specification Reccomendations

IETF 99
July 2017
(draft-bertz-dime-diamimpr-00)

Motivation

- As an Operator, Sprint was observing successful interop of Diameter but long lead times for development
- Sprint's C3PO project (a 4G open source EPC developed jointly with Intel) uses freediameter
  - Also noticed longer than expected implementation lead times
  - Errors were minor but enough to be a drag on the schedule
  - We decided to do a study to determine how efficient we can be
- Hypothesis – We can go from spec to code (library + structs / objects + json marshaling +...) < 1 hour

Findings

- We were so wrong...
- Too many table formats
  - It cannot be argued this is a Change over time, some specs started / became RFCs at similar times but have different Table formats
- Imports of AVPs not treated consistently
- Outright errors slow us down
  - Some AVPs NEVER given a Code
  - Some AVPs referenced and missing (NEVER defined)
- Some data you just can't get from any spec programmatically
  - App ID + Messages + Return Codes + App Name
  - Especially hard when a spec has multiple applications

Methodology

- Save Document in text format
- Put Defined AVPS and Imported/Referenced/Re-used AVPs in different files
- Add Enums ONLY if parser cannot find them
- Put app info in a special file to map commands, return codes, app id, app name, etc.
- Run python code to create 'dia' files used in diafuzzer (https://github.com/Orange-OpenSource/diafuzzer)
  - Enhance as required to support use cases
- Load the dia files into memory (which forces many semantic checks)

Unexpected Use Cases

- Grouped AVPs/Commands that further refine the AVPs/Commands defined in other applications
- Easy to detect
  - Except when in the same document (that was no fun)
- Some inconsistencies - in some specs the Grouped AVP was Imported and others it was defined when in fact it is both
- There is no concept of imported commands...

We are not immune

- RFC 4006 bis had a few minor issues (to be closed as part of WGLC)

Enumerations

- Don't use them. Just don't...
- Use Cases
  - Enums referencing other Enums
    - Some reuse Enums
    - Others restrict Enums – can you do that?
    - Others add new values
  - Enums referencing web pages (URIs)
  - Enums reference registries (which is expected)
- No format and very inconsistent
  - So inconsistent we often just appended enums to generated files and manually processed them

Recommendations

- Overall:
  - The name of all AVPs, Commands and Grouped AVPs appear consistently throughout the document.
  - The letter case MUST be consistent for all names.
  - No spaces should appear in the names.
  - Use of underscores is discouraged except for line continuations in tables and enumeration labels.

Defined AVP Recommendations

- Tables MUST include the following columns:
  - Attribute Name
  - AVP Code
  - Section Defined
  - Data Type
  - AVP Flag Rules for MUST and MUST NOT
  - Other minor recommendations in spec

Defined AVPs Recommended Formats

Attribute Name	AVP Code	Section Defined	Data Type	MUST	MUST NOT	
AVP-Name	85	9.8.2	Unsigned32	M	V	

Example Two

Attribute Name	AVP Code	Section Defined	Data Type	MUST	MUST NOT	
AVP-Name	85	9.8.2	Unsigned32	M	V	

Imported AVP Recommendations

- Imported or Re-used AVPs MUST be included in the specification. A table MUST be present if AVPs are re-used/imported.
  - The table MUST include the AVP and Source document columns.
  - The table MAY include a Comment column.
  - An M-bit column MAY be present as required.
    - (Should this be MAY or MUST?)
  - The table MUST be pipe delimited when in text format.

Grouped AVP / Command Refinement

- Preconditions
  - Original structure MUST have *[AVP] present to be Refinable (see Open Questions later in presentation about this)
  - Any refinement of an AVP present the new structure MUST conform to the occurrences range
    - If X*Y[Foo] was present in original and A*B [Foo] present in refinement then X <= A <= B <= Y.
  - AVPs retained without further restriction of the number of occurrences MUST be kept in the Refining AVP's definition otherwise they are assumed to be dropped from the new AVPdefinition.
- Refinement is defined by 'Refines [App Id]'. When App Id is not present, it is assumed to be the original spec.
  - A Refined AVP / Command MUST NOT appear in anything but an Import Table EXCEPT for the specification that originally defined the AVP / Command.
    - Allows for easy detection of defining spec but not helpful when a define/refine is in the same document.<< Open question

Example Refinement

From TS 29.336

User-Identifier ::= <AVP-Header: 3102, 10415>
[User-Name]
[MSISDN]
[External-Identifier]
[LMSI]
*[AVP]

From TS 29.128

User-Identifier ::= <AVP-Header: 3102, 10415, **Refines**>
[User-Name]
*[AVP]

Command Recommendations

- inconsistent values between the name, three letter acronym defined in the table and the actual name used in the command definition.

Enumeration Recommendations

- Enumeration Value Names MUST adhere to alphanumeric and underscore characters.
- Enumeration Value Names MUST not begin with an underscore.
- When being defined the format MUST include the label and the value assigned with the label enclosed in parenthesis on a single line.
  - Example w/o parenthesis
    - Speed_10 10
    - Speed_1010 (Error)
  - Example with parenthesis
    - Speed_10 (10)
    - Speed_10(10)
- Coverage of Use Cases were NOT specified in the document

GAPs for Automated Validation

Many pieces of information cannot be programmatically validated.

GAP 1: The application identifier and name of an application.

GAP 2: The application and vendor identifiers associated with a defined AVP table.

GAP 3: The application and vendor identifiers associated with Commands.

GAP 4: Reused and newly defined result codes for an application.

GAP 5: Easily parsed enumerations that cover all use cases.

Example to Close Gaps

- Just a suggestion here

```
1: AppFoo ::= <Diameter Application: 10415 101010>
2:  Command1-Name-Request C1R
3:  Command1-Name-Answer C1A
4:
5: Result-Codes ::= <Diameter Result-Codes: 101010>
6:  NEW_RESULT (4999)
7:  IMPORTED_RESULT IMPORT (4010)
```

GAP 1 is closed in line 1. GAP 3 is closed in lines 1 through 3 while GAP 4 is closed by lines 5 through 7.

GAP 2 can be closed by using a common discernable Table Name format, e.g. AppFoo defined AVPs. In this case the Application Name can be looked up and associated to the defined AVP table.

Enumeration Example Format

Gap 5 can be partially closed by following a pattern similar to Result-Codes but this does not resolve all uses cases.

```
Result-Codes ::= <Diameter Enumeration: 123, 45678>
   Label_1 (0)
   LABEL_Two (2)
```

Does not resolve Use Cases.

Open Questions

- Open question, can a Grouped AVP/Command have a range limited [AVP] member, e.g. *5[AVP]?
- Do we go back and file Errata on all items noted in the Survey?
  - Will we take up fixing these?
- Do we want to mix 'pseudo enumerations' (defined as Unsigned32 but have a spec for mapping labels to values and DOES NOT require a registry)
- When a define/refine is in the same document we know it should be in a Defined Grouped AVP (if it is a Grouped AVP) but should we make other considerations?

Summary

- Diameter over the wire is fine
- The specs are doing well (our evidence is that over the wire interoperability works)
- We can do better!
- Our current formats cannot cover all use cases
- Automated Validation is possible
- Good specs take less than 20 minutes to go from spec to code generation

Next Steps

- WG Adopt this document as either

  A> a requirements document and make changes in another document (my recommendation)

  B> a solutions document (much more work required)
- File Errata based upon results but don't work on fixes unless the specs are actually used
- Figure out what to do for Enums (add as WG Item)
- Work on an automated verification system as part of the I-D nits process
  - Should it be gating all of the time or only for WGLC?