Internet Engineering Task Force (IETF)
Request for Comments: 7423
BCP: 193
Category: Best Current Practice
ISSN: 2070-1721

L. Morand, Ed.
Orange Labs
V. Fajardo
Fluke Networks
H. Tschofenig

November 2014

Diameter Applications Design Guidelines

Abstract

The Diameter base protocol provides facilities for protocol extensibility enabling the definition of new Diameter applications or modification of existing applications. This document is a companion document to the Diameter base protocol that further explains and clarifies the rules to extend Diameter. Furthermore, this document provides guidelines to Diameter application designers reusing/defining Diameter applications or creating generic Diameter extensions.

Status of This Memo

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7423.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction

The Diameter base protocol [RFC6733] is intended to provide an Authentication, Authorization, and Accounting (AAA) framework for applications such as network access or IP mobility in both local and roaming situations. This protocol provides the ability for Diameter peers to exchange messages carrying data in the form of Attribute-Value Pairs (AVPs).

The Diameter base protocol provides facilities to extend Diameter (see Section 1.3 of [RFC6733]) to support new functionality. In the context of this document, extending Diameter means one of the following:

1. The addition of new functionality to an existing Diameter application without defining a new application.

2. The addition of new functionality to an existing Diameter application that requires the definition of a new application.

3. The definition of an entirely new Diameter application to offer functionality not supported by existing applications.

4. The definition of a new generic functionality that can be reused across different applications.

All of these extensions are design decisions that can be carried out by any combination of reusing existing or defining new commands, AVPs, or AVP values. However, application designers do not have complete freedom when making their design. A number of rules have been defined in [RFC6733] that place constraints on when an extension requires the allocation of a new Diameter application identifier or a new command code value. The objective of this document is the following:

- Clarify the Diameter extensibility rules as defined in the Diameter base protocol.

- Discuss design choices and provide guidelines when defining new applications.

- Present trade-off choices.

2. Terminology

This document reuses the terminology defined in [RFC6733]. Additionally, the following terms and acronyms are used in this application:

Application: Extension of the Diameter base protocol [RFC6733] via the addition of new commands or AVPs. Each application is uniquely identified by an IANA-allocated application identifier value.

Command: Diameter request or answer carrying AVPs between Diameter endpoints. Each command is uniquely identified by an IANA-allocated Command Code value and is described by a Command Code Format (CCF) for an application.

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 [RFC2119].

3. Overview

As designed, the Diameter base protocol [RFC6733] can be seen as a two-layer protocol. The lower layer is mainly responsible for managing connections between neighboring peers and for message routing. The upper layer is where the Diameter applications reside. This model is in line with a Diameter node having an application layer and a peer-to-peer delivery layer. The Diameter base protocol document defines the architecture and behavior of the message delivery layer and then provides the framework for designing Diameter applications on the application layer. This framework includes definitions of application sessions and accounting support (see Sections 8 and 9 of [RFC6733]). Accordingly, a Diameter node is seen in this document as a single instance of a Diameter message delivery layer and one or more Diameter applications using it.

The Diameter base protocol is designed to be extensible and the principles are described in Section 1.3 of [RFC6733]. In summary, Diameter can be extended by the following:

1. Defining new AVP values
2. Creating new AVPs
3. Creating new commands
4. Creating new applications

As a main guiding principle, application designers SHOULD comply with the following recommendation: "try to reuse as much as possible!". It will reduce the time to finalize specification writing, and it will lead to a smaller implementation effort as well as reduce the need for testing. In general, it is clever to avoid duplicate effort when possible.

However, reuse is not appropriate when the existing functionality does not fit the new requirement and/or the reuse leads to ambiguity.

The impact on extending existing applications can be categorized into two groups:

Minor Extension: Enhancing the functional scope of an existing application by the addition of optional features to support it. Such enhancement has no backward-compatibility issue with the existing application.

A typical example would be the definition of a new optional AVP for use in an existing command. Diameter implementations supporting the existing application but not the new AVP will simply ignore it, without consequences for the Diameter message handling, as described in [RFC6733]. The standardization effort will be fairly small.

Major Extension: Enhancing an application that requires the definition of a new Diameter application. Such enhancement causes a backward-compatibility issue with existing implementations supporting the application.

Typical examples would be the creation of a new command for providing functionality not supported by existing applications or the definition of a new AVP to be carried in an existing command with the M-bit set in the AVP flags (see Section 4.1 of [RFC6733] for definition of "M-bit"). For such an extension, a significant specification effort is required, and a careful approach is recommended.

4. Reusing Existing Diameter Applications

An existing application may need to be enhanced to fulfill new requirements, and these modifications can be at the command level and/or at the AVP level. The following sections describe the possible modifications that can be performed on existing applications and their related impact.

4.1. Adding a New Command

Adding a new command to an existing application is considered to be a major extension and requires a new Diameter application to be defined, as stated in Section 1.3.4 of [RFC6733]. The need for a new application is because a Diameter node that is not upgraded to support the new command(s) within the (existing) application would reject any unknown command with the protocol error DIAMETER_COMMAND_UNSUPPORTED and cause the failure of the transaction. The new application ensures that Diameter nodes only receive commands within the context of applications they support.

Adding a new command means either defining a completely new command or importing the command's Command Code Format (CCF) syntax from another application whereby the new application inherits some or all of the functionality of the application from which the command came. In the former case, the decision to create a new application is straightforward, since this is typically a result of adding a new functionality that does not exist yet. For the latter, the decision to create a new application will depend on whether importing the command in a new application is more suitable than simply using the existing application as it is in conjunction with any other application.

An example considers the Diameter Extensible Authentication Protocol (EAP) application [RFC4072] and the Diameter Network Access Server application [RFC7155]. When network access authentication using EAP is required, the Diameter EAP commands (Diameter-EAP-Request/ Diameter-EAP-Answer) are used; otherwise, the Diameter Network Access Server application will be used. When the Diameter EAP application is used, the accounting exchanges defined in the Diameter Network Access Server may be used.

However, in general, it is difficult to come to a hard guideline, and so a case-by-case study of each application requirement should be applied. Before adding or importing a command, application designers should consider the following:

- Can the new functionality be fulfilled by creating a new command independent from any existing command? In this case, the resulting new application and the existing application can work independent of, but cooperating with, each other.

- Can the existing command be reused without major extensions and, therefore, without the need for the definition of a new application, e.g., new functionality introduced by the creation of new optional AVPs.

It is important to note that importing commands too liberally could result in a monolithic and hard-to-manage application supporting too many different features.

4.2. Deleting an Existing Command

Although this process is not typical, removing a command from an application requires a new Diameter application to be defined, and then it is considered as a major extension. This is due to the fact that the reception of the deleted command would systematically result in a protocol error (i.e., DIAMETER_COMMAND_UNSUPPORTED).

It is unusual to delete an existing command from an application for the sake of deleting it or the functionality it represents. An exception might be if the intent of the deletion is to create a newer variance of the same application that is somehow simpler than the application initially specified.

4.3. Reusing Existing Commands

This section discusses rules in adding and/or deleting AVPs from an existing command of an existing application. The cases described in this section may not necessarily result in the creation of new applications.

From a historical point of view, it is worth noting that there was a strong recommendation to reuse existing commands in [RFC3588] to prevent rapid depletion of code values available for vendor-specific commands. However, [RFC6733] has relaxed the allocation policy and enlarged the range of available code values for vendor-specific applications. Although reuse of existing commands is still RECOMMENDED, protocol designers can consider defining a new command when it provides a solution more suitable than the twisting of an existing command’s use and applications.

4.3.1. Adding AVPs to a Command

Based on the rules in [RFC6733], AVPs that are added to an existing command can be categorized as either:

- Mandatory (to understand) AVPs. As defined in [RFC6733], these are AVPs with the M-bit flag set in this command, which means that the Diameter node receiving them is required to understand not only their values but also their semantics. Failure to do so will cause a message handling error: either an error message with the result-code set to DIAMETER_AVP_UNSUPPORTED if the AVP is not understood in a request or an application-specific error handling if the given AVP is in an answer.

- Optional (to understand) AVPs. As defined in [RFC6733], these are AVPs with the M-bit flag cleared in this command. A Diameter node receiving these AVPs can simply ignore them if it does not support them.

It is important to note that the definitions given above are independent of whether these AVPs are required or optional in the command as specified by the command's CCF syntax [RFC6733].

> NOTE: As stated in [RFC6733], the M-bit setting for a given AVP is relevant to an application and each command within that application that includes the AVP.

The rules are strict in the case where the AVPs to be added in an exiting command are mandatory to understand, i.e., they have the M-bit set. A mandatory AVP MUST NOT be added to an existing command without defining a new Diameter application, as stated in [RFC6733]. This falls into the "Major Extensions" category. Despite the clarity of the rule, ambiguity still arises when evaluating whether a new AVP being added should be mandatory to begin with. Application designers should consider the following questions when deciding about the M-bit for a new AVP:

- Would it be required for the receiving side to be able to process and understand the AVP and its content?
- Would the new AVPs change the state machine of the application?
- Would the presence of the new AVP lead to a different number of round trips, effectively changing the state machine of the application?
- Would the new AVP be used to differentiate between old and new variances of the same application whereby the two variances are not backward compatible?
- Would the new AVP have duality in meaning, i.e., be used to carry application-related information as well as to indicate that the message is for a new application?

If the answer to at least one of the questions is "yes", then the M-bit MUST be set for the new AVP, and a new Diameter application MUST be defined. This list of questions is non-exhaustive, and other criteria MAY be taken into account in the decision process.

If application designers are instead contemplating the use of optional AVPs, i.e., with the M-bit cleared, there are still pitfalls that will cause interoperability problems; therefore, they must be avoided. Some examples of these pitfalls are as follows:

- Use of optional AVPs with intersecting meaning. One AVP has partially the same usage and meaning as another AVP. The presence of both can lead to confusion.

- Optional AVPs with dual purpose, i.e., to carry application data as well as to indicate support for one or more features. This has a tendency to introduce interpretation issues.

- Adding one or more optional AVPs and indicating (usually within descriptive text for the command) that at least one of them has to be understood by the receiver of the command. This would be equivalent to adding a mandatory AVP, i.e., an AVP with the M-bit set, to the command.

4.3.2. Deleting AVPs from a Command

Application designers may want to reuse an existing command, but some of the AVPs present in the command's CCF syntax specification may be irrelevant for the functionality foreseen to be supported by this command. It may be then tempting to delete those AVPs from the command.

The impacts of deleting an AVP from a command depends on its Command Code format specification and M-bit setting:

- Case 1: Deleting an AVP that is indicated as a required AVP (noted as {AVP}) in the command's CCF syntax specification (regardless of the M-bit setting).

  In this case, a new Command Code, and subsequently a new Diameter application, MUST be specified.

- Case 2: Deleting an AVP, which has the M-bit set, and is indicated as an optional AVP (noted as [AVP] in the command CCF) in the command's CCF syntax specification.

  In this case, no new Command Code has to be specified, but the definition of a new Diameter application is REQUIRED.

- Case 3: Deleting an AVP, which has the M-bit cleared, and is indicated as [AVP] in the command's CCF syntax specification.

  In this case, the AVP can be deleted without consequences.

Application designers SHOULD attempt to reuse the command's CCF syntax specification without modification and simply ignore (but not delete) any optional AVPs that will not be used. This is to maintain compatibility with existing applications that will not know about the new functionality as well as to maintain the integrity of existing dictionaries.

4.3.3. Changing the Flag Settings of AVP in Existing Commands

Although unusual, implementors may want to change the setting of the AVP flags a given AVP used in a command.

Into an existing command, an AVP that was initially defined as a mandatory AVP to understand, i.e., an AVP with the M-bit flag set in the command MAY be safely turned to an optional AVP, i.e., with the M-bit cleared. Any node supporting the existing application will still understand the AVP, whatever the setting of the M-bit. On the contrary, an AVP initially defined as an optional AVP to understand, i.e., an AVP with the M-bit flag cleared in the command MUST NOT be changed into a mandatory AVP with the M-bit flag set without defining a new Diameter application. Setting the M-bit for an AVP that was defined as an optional AVP is equivalent to adding a new mandatory AVP to an existing command, and the rules given in Section 4.3.1 apply.

All other AVP flags (V-bit, P-bit, reserved bits) MUST remain unchanged.

4.4. Reusing Existing AVPs

This section discusses rules in reusing existing AVPs when reusing an existing command or defining a new command in a new application.

4.4.1. Setting of the AVP Flags

When reusing existing AVPs in a new application, application designers MUST specify the setting of the M-bit flag for a new Diameter application and, if necessary, for every command of the application that can carry these AVPs. In general, for AVPs defined outside of the Diameter base protocol, the characteristics of an AVP are tied to its role within a given application and the commands used in this application.

All other AVP flags (V-bit, P-bit, reserved bits) MUST remain unchanged.

4.4.2. Reuse of AVP of Type Enumerated

When reusing an AVP of type Enumerated in a command for a new application, it is RECOMMENDED to avoid modifying the set of valid values defined for this AVP. Modifying the set of Enumerated values includes adding a value or deprecating the use of a value defined initially for the AVP. Modifying the set of values will impact the application defining this AVP and all the applications using this AVP, causing potential interoperability issues: a value used by a peer that will not be recognized by all the nodes between the client and the server will cause an error response with the Result-Code AVP set to DIAMETER_INVALID_AVP_VALUE. When the full range of values defined for this Enumerated AVP is not suitable for the new application, it is RECOMMENDED that a new AVP be defined to avoid backward-compatibility issues with existing implementations.

5. Defining New Diameter Applications

5.1. Introduction

This section discusses the case where new applications have requirements that cannot be fulfilled by existing applications and would require definition of completely new commands, AVPs, and/or AVP values. Typically, there is little ambiguity about the decision to create these types of applications. Some examples are the interfaces defined for the IP Multimedia Subsystem of 3GPP, e.g., Cx/Dx ([TS29.228] and [TS29.229]), Sh ([TS29.328] and [TS29.329]), etc.

Application designers SHOULD try to import existing AVPs and AVP values for any newly defined commands. In certain cases where accounting will be used, the models described in Section 5.10 SHOULD also be considered.

Additional considerations are described in the following sections.

5.2. Defining New Commands

As a general recommendation, commands SHOULD NOT be defined from scratch. It is instead RECOMMENDED to reuse an existing command offering similar functionality and use it as a starting point. Code reuse leads to a smaller implementation effort as well as reduces the need for testing.

Moreover, the new command’s CCF syntax specification SHOULD be carefully defined when considering applicability and extensibility of the application. If most of the AVPs contained in the command are indicated as fixed or required, it might be difficult to reuse the same command and, therefore, the same application in a slightly

changed environment. Defining a command with most of the AVPs indicated as optional is considered as a good design choice in many cases, despite the flexibility it introduces in the protocol. Protocol designers MUST clearly state the reasons why these optional AVPs might or might not be present and properly define the corresponding behavior of the Diameter nodes when these AVPs are absent from the command.

> NOTE: As a hint for protocol designers, it is not sufficient to just look at the command's CCF syntax specification. It is also necessary to carefully read through the accompanying text in the specification.

In the same way, the CCF syntax specification SHOULD be defined such that it will be possible to add any arbitrary optional AVPs with the M-bit cleared (including vendor-specific AVPs) without modifying the application. For this purpose, "* [AVP]" SHOULD be added in the command's CCF, which allows the addition of any arbitrary number of optional AVPs as described in [RFC6733].

5.3. Use of Application Id in a Message

When designing new applications, application designers SHOULD specify that the Application Id carried in all session-level messages is the Application Id of the application using those messages. This includes the session-level messages defined in the Diameter base protocol, i.e., Re-Auth-Request (RAR) / Re-Auth-Answer (RAA), Session-Termination-Request (STR) / Session-Termination-Answer (STA), Abort-Session-Request (ASR) / Abort-Session-Answer (ASA), and possibly Accounting-Request (ACR) / Accounting Answer (ACA) in the coupled accounting model; see Section 5.10. Some existing specifications do not adhere to this rule for historical reasons. However, this guidance SHOULD be followed by new applications to avoid routing problems.

When a new application has been allocated with a new Application Id and it also reuses existing commands with or without modifications, the commands SHOULD use the newly allocated Application Id in the header and in all relevant Application-Id AVPs (Auth-Application-Id or Acct-Application-Id) present in the commands message body.

Additionally, application designers using a vendor-specific Application-Id AVP SHOULD NOT use the Vendor-Id AVP to further dissect or differentiate the vendor-specification Application Id. Diameter routing is not based on the Vendor Id. As such, the Vendor Id SHOULD NOT be used as an additional input for routing or delivery of messages. The Vendor-Id AVP is an informational AVP only and kept for backward compatibility reasons.

5.4. Application-Specific Session State Machines

Section 8 of [RFC6733] provides session state machines for AAA services, and these session state machines are not intended to cover behavior outside of AAA. If a new application cannot clearly be categorized into any of these AAA services, it is RECOMMENDED that the application define its own session state machine. Support for a server-initiated request is a clear example where an application-specific session state machine would be needed, for example, the Rw interface for the ITU-T push model (cf. [Q.3303.3]).

5.5. Session-Id AVP and Session Management

Diameter applications are usually designed with the aim of managing user sessions (e.g., Diameter Network Access Server (NAS) application [RFC4005]) or a specific service access session (e.g., Diameter SIP application [RFC4740]). In the Diameter base protocol, session state is referenced using the Session-Id AVP. All Diameter messages that use the same Session-Id will be bound to the same session. Diameter-based session management also implies that both the Diameter client and server (and potentially proxy agents along the path) maintain session state information.

However, some applications may not need to rely on the Session-Id to identify and manage sessions because other information can be used instead to correlate Diameter messages. Indeed, the User-Name AVP or any other specific AVP can be present in every Diameter message and used, therefore, for message correlation. Some applications might not require the notion of the Diameter-session concept at all. For such applications, the Auth-Session-State AVP is usually set to NO_STATE_MAINTAINED in all Diameter messages, and these applications are, therefore, designed as a set of stand-alone transactions. Even if an explicit access session termination is required, application-specific commands are defined and used instead of the STR/STA or ASR/ASA defined in the Diameter base protocol [RFC6733]. In such a case, the Session-Id is not significant.

Based on these considerations, protocol designers should carefully appraise whether the Diameter application being defined relies on the session management specified in the Diameter base protocol:

- If it is, the Diameter command defined for the new application MUST include the Session-Id AVP defined in the Diameter base protocol [RFC6733], and the Session-Id AVP MUST be used for correlation of messages related to the same session. Guidance on the use of the Auth-Session-State AVP is given in the Diameter base protocol [RFC6733].

- Otherwise, because session management is not required or the application relies on its own session management mechanism, Diameter commands for the application need not include the Session-Id AVP. If any specific session management concept is supported by the application, the application documentation MUST clearly specify how the session is handled between the client and server (and possibly Diameter agents in the path). Moreover, because the application is not maintaining session state at the Diameter base protocol level, the Auth-Session-State AVP MUST be included in all Diameter commands for the application and MUST be set to NO_STATE_MAINTAINED.

5.6. Use of Enumerated Type AVPs

The type Enumerated was initially defined to provide a list of valid values for an AVP with their respective interpretation described in the specification. For instance, AVPs of type Enumerated can be used to provide further information on the reason for the termination of a session or a specific action to perform upon the reception of the request.

As described in Section 4.4.2 above, defining an AVP of type Enumerated presents some limitations in terms of extensibility and reusability. Indeed, the finite set of valid values defined in the definition of the AVP of type Enumerated cannot be modified in practice without causing backward-compatibility issues with existing implementations. As a consequence, AVPs of type Enumerated MUST NOT be extended by adding new values to support new capabilities. Diameter protocol designers SHOULD carefully consider before defining an Enumerated AVP whether the set of values will remain unchanged or new values may be required in the near future. If such an extension is foreseen or cannot be avoided, it is RECOMMENDED to define AVPs of type Unsigned32 or Unsigned64 in which the data field would contain an address space representing "values" that would have the same use of Enumerated values. Whereas only the initial values defined at the definition of the AVP of type Enumerated are valid as described in Section 4.4.2, any value from the address space from 0 to 2^32 - 1 for AVPs of type Unsigned32 or from 0 to 2^64 - 1 for AVPs of type Unsigned64 is valid at the Diameter base protocol level and will not cause interoperability issues for intermediary nodes between clients and servers. Only clients and servers will be able to process the values at the application layer.

For illustration, an AVP describing possible access networks would be defined as follows:

Access-Network-Type AVP (XXX) is of type Unsigned32 and contains a 32-bit address space representing types of access networks. This application defines the following classes of access networks, all identified by the thousands digit in the decimal notation:

- 1xxx (Mobile Access Networks)
- 2xxx (Fixed Access Networks)
- 3xxx (Wireless Access Networks)

Values that fall within the Mobile Access Networks category are used to inform a peer that a request has been sent for a user attached to a mobile access network. The following values are defined in this application:

1001: 3GPP-GERAN

The user is attached to a Global System for Mobile Communications (GSM) Enhanced Data rates for GSM Evolution (EDGE) Radio Access Network.

1002: 3GPP-UTRAN-FDD

The user is attached to a Universal Mobile Telecommunications System (UMTS) access network that uses frequency-division duplexing for duplexing.

Unlike Enumerated AVP, any new value can be added in the address space defined by this Unsigned32 AVP without modifying the definition of the AVP. There is, therefore, no risk of backward-compatibility issues, especially when intermediate nodes may be present between Diameter endpoints.

Along the same line, AVPs of type Enumerated are too often used as a simple Boolean flag, indicating, for instance, a specific permission or capability; therefore, only three values are defined, e.g., TRUE/FALSE, AUTHORIZED/UNAUTHORIZED, or SUPPORTED/UNSUPPORTED. This is a sub-optimal design since it limits the extensibility of the application: any new capability/permission would have to be supported by a new AVP or new Enumerated value of the already-defined AVP, with the backward-compatibility issues described above. Instead of using an Enumerated AVP for a Boolean flag, protocol designers SHOULD use AVPs of type Unsigned32 or Unsigned64 in which the data field would

be defined as a bit mask whose bit settings are described in the relevant Diameter application specification. Such AVPs can be reused and extended without major impact on the Diameter application. The bit mask SHOULD leave room for future additions. Examples of AVPs that use bit masks are the Session-Binding AVP defined in [RFC6733] and the MIP6-Feature-Vector AVP defined in [RFC5447].

5.7. Application-Specific Message Routing

As described in [RFC6733], a Diameter request that needs to be sent to a home server serving a specific realm, but not to a specific server (such as the first request of a series of round trips), will contain a Destination-Realm AVP and no Destination-Host AVP.

For such a request, the message routing usually relies only on the Destination-Realm AVP and the Application Id present in the request message header. However, some applications may need to rely on the User-Name AVP or any other application-specific AVPs present in the request to determine the final destination of a request, e.g., to find the target AAA server hosting the authorization information for a given user when multiple AAA servers are addressable in the realm.

In such a context, basic routing mechanisms described in [RFC6733] are not fully suitable, and additional application-level routing mechanisms MUST be described in the application documentation to provide such specific AVP-based routing. Such functionality will be basically hosted by an application-specific proxy agent that will be responsible for routing decisions based on the received specific AVPs.

Examples of such application-specific routing functions can be found in the Cx/Dx applications ([TS29.228] and [TS29.229]) of the 3GPP IP Multimedia Subsystem, in which the proxy agent (Subscriber Location Function, aka SLF) uses specific application-level identities found in the request to determine the final destination of the message.

Whatever the criteria used to establish the routing path of the request, the routing of the answer MUST follow the reverse path of the request, as described in [RFC6733], with the answer being sent to the source of the received request, using transaction states and hop-by-hop identifier matching. This ensures that the Diameter relay or proxy agents in the request routing path will be able to release the transaction state upon receipt of the corresponding answer, avoiding unnecessary failover. Moreover, especially in roaming cases, proxy agents in the path must be able to apply local policies when receiving the answer from the server during authentication/ authorization and/or accounting procedures and maintain up-to-date session state information by keeping track of all authorized active

sessions. Therefore, application designers MUST NOT modify the answer-routing principles described in [RFC6733] when defining a new application.

5.8. Translation Agents

As defined in [RFC6733], a translation agent is a device that provides interworking between Diameter and another AAA protocol, such as RADIUS.

In the case of RADIUS, it was initially thought that defining the translation function would be straightforward by adopting a few basic principles, e.g., by the use of a shared range of code values for RADIUS attributes and Diameter AVPs. Guidelines for implementing a RADIUS-Diameter translation agent were put into the Diameter NAS Application [RFC4005].

However, it was acknowledged that such a translation mechanism was not so obvious and deeper protocol analysis was required to ensure efficient interworking between RADIUS and Diameter. Moreover, the interworking requirements depend on the functionalities provided by the Diameter application under specification, and a case-by-case analysis is required. As a consequence, all the material related to RADIUS-to-Diameter translation is removed from the new version of the Diameter NAS Application specification [RFC7155], which deprecates RFC 4005 [RFC4005].

Therefore, protocol designers SHOULD NOT assume the availability of a "standard" Diameter-to-RADIUS gateway agent when planning to interoperate with the RADIUS infrastructure. They SHOULD specify the required translation mechanism along with the Diameter application, if needed. This recommendation applies for any kind of translation.

5.9. End-to-End Application Capabilities Exchange

Diameter applications can rely on optional AVPs to exchange application-specific capabilities and features. These AVPs can be exchanged on an end-to-end basis at the application layer. Examples of this can be found with the MIP6-Feature-Vector AVP in [RFC5447] and the QoS-Capability AVP in [RFC5777].

End-to-end capabilities AVPs can be added as optional AVPs with the M-bit cleared to existing applications to announce support of new functionality. Receivers that do not understand these AVPs or the AVP values can simply ignore them, as stated in [RFC6733]. When supported, receivers of these AVPs can discover the additional functionality supported by the Diameter endpoint originating the request and behave accordingly when processing the request. Senders

of these AVPs can safely assume the receiving endpoint does not support any functionality carried by the AVP if it is not present in the corresponding response. This is useful in cases where deployment choices are offered, and the generic design can be made available for a number of applications.

When used in a new application, these end-to-end capabilities AVPs SHOULD be added as an optional AVP into the CCF of the commands used by the new application. Protocol designers SHOULD clearly specify this end-to-end capabilities exchange and the corresponding behavior of the Diameter nodes supporting the application.

It is also important to note that this end-to-end capabilities exchange relying on the use of optional AVPs is not meant as a generic mechanism to support extensibility of Diameter applications with arbitrary functionality. When the added features drastically change the Diameter application or when Diameter agents must be upgraded to support the new features, a new application SHOULD be defined, as recommended in [RFC6733].

5.10. Diameter Accounting Support

Accounting can be treated as an auxiliary application that is used in support of other applications. In most cases, accounting support is required when defining new applications. This document provides two possible models for using accounting:

Split Accounting Model:

In this model, the accounting messages will use the Diameter base accounting Application Id (value of 3). The design implication for this is that the accounting is treated as an independent application, especially for Diameter routing. This means that accounting commands emanating from an application may be routed separately from the rest of the other application messages. This may also imply that the messages end up in a central accounting server. A split accounting model is a good design choice when:

* The application itself does not define its own accounting commands.

* The overall system architecture permits the use of centralized accounting for one or more Diameter applications.

Centralizing accounting may have advantages, but there are also drawbacks. The model assumes that the accounting server can differentiate received accounting messages. Since the received accounting messages can be for any application and/or service, the

accounting server MUST have a method to match accounting messages with applications and/or services being accounted for. This may mean defining new AVPs; checking the presence, absence, or contents of existing AVPs; or checking the contents of the accounting record itself. One of these means could be to insert into the request sent to the accounting server an Auth-Application-Id AVP containing the identifier of the application for which the accounting request is sent. But in general, there is no clean and generic scheme for sorting these messages. Therefore, this model SHOULD NOT be used when all received accounting messages cannot be clearly identified and sorted. For most cases, the use of the Coupled Accounting Model is RECOMMENDED.

Coupled Accounting Model:

In this model, the accounting messages will use the Application Id of the application using the accounting service. The design implication for this is that the accounting messages are tightly coupled with the application itself, meaning that accounting messages will be routed like the other application messages. It would then be the responsibility of the application server (application entity receiving the ACR message) to send the accounting records carried by the accounting messages to the proper accounting server. The application server is also responsible for formulating a proper response (ACA). A coupled accounting model is a good design choice when:

* The system architecture or deployment does not provide an accounting server that supports Diameter. Consequently, the application server MUST be provisioned to use a different protocol to access the accounting server, e.g., via the Lightweight Directory Access Protocol (LDAP), SOAP, etc. This case includes the support of older accounting systems that are not Diameter aware.

* The system architecture or deployment requires that the accounting service for the specific application should be handled by the application itself.

In all cases above, there will generally be no direct Diameter access to the accounting server.

These models provide a basis for using accounting messages. Application designers may obviously deviate from these models provided that the factors being addressed here have also been taken

into account. As a general recommendation, application designers SHOULD NOT define a new set of commands to carry application-specific accounting records.

5.11. Diameter Security Mechanisms

As specified in [RFC6733], the Diameter message exchange SHOULD be secured between neighboring Diameter peers using Transport Layer Security (TLS) / TCP or Datagram Transport Layer Security (DTLS) / Stream Control Transmission Protocol (SCTP). However, IPsec MAY also be deployed to secure communication between Diameter peers. When IPsec is used instead of TLS or DTLS, the following recommendations apply.

IPsec Encapsulating Security Payload (ESP) [RFC4301] in transport mode with non-null encryption and authentication algorithms MUST be used to provide per-packet authentication, integrity protection, and confidentiality and to support the replay protection mechanisms of IPsec. Internet Key Exchange Protocol Version 2 (IKEv2) [RFC7296] SHOULD be used for performing mutual authentication and for establishing and maintaining security associations (SAs).

Version 1 of IKE (IKEv1), defined in [RFC2409], was initially used for peer authentication, negotiation of security associations, and key management in RFC 3588 [RFC3588]. For easier migration from the obsoleted implementations based on IKEv1 to IKEv2, both RSA digital signatures and pre-shared keys SHOULD be supported in IKEv2. However, if IKEv1 is used, implementors SHOULD follow the guidelines given in Section 13.1 of RFC 3588 [RFC3588].

6. Defining Generic Diameter Extensions

Generic Diameter extensions are AVPs, commands, or applications that are designed to support other Diameter applications. They are auxiliary applications meant to improve or enhance the Diameter protocol itself or Diameter applications/functionality. Some examples include the extensions to support realm-based redirection of Diameter requests (see [RFC7075]), conveying a specific set of priority parameters influencing the distribution of resources (see [RFC6735]), and the support for QoS AVPs (see [RFC5777]).

Since generic extensions may cover many aspects of Diameter and Diameter applications, it is not possible to enumerate all scenarios. However, some of the most common considerations are as follows:

Backward Compatibility:

When defining generic extensions designed to be supported by existing Diameter applications, protocol designers MUST consider the potential impacts of the introduction of the new extension on the behavior of the node that would not be yet upgraded to support/understand this new extension. Designers MUST also ensure that new extensions do not break expected message delivery layer behavior.

Forward Compatibility:

Protocol designers MUST ensure that their design will not introduce undue restrictions for future applications.

Trade-off in Signaling:

Designers may have to choose between the use of optional AVPs piggybacked onto existing commands versus defining new commands and applications. Optional AVPs are simpler to implement and may not need changes to existing applications. However, this ties the sending of extension data to the application's transmission of a message. This has consequences if the application and the extensions have different timing requirements. The use of commands and applications solves this issue, but the trade-off is the additional complexity of defining and deploying a new application. It is left up to the designer to find a good balance among these trade-offs based on the requirements of the extension.

In practice, generic extensions often use optional AVPs because they are simple and non-intrusive to the application that would carry them. Peers that do not support the generic extensions need not understand nor recognize these optional AVPs. However, it is RECOMMENDED that the authors of the extension specify the context or usage of the optional AVPs. As an example, in the case that the AVP can be used only by a specific set of applications, then the specification MUST enumerate these applications and the scenarios when the optional AVPs will be used. In the case where the optional AVPs can be carried by any application, it should be sufficient to specify such a use case and perhaps provide specific examples of applications using them.

In most cases, these optional AVPs piggybacked by applications would be defined as a Grouped AVP, and it would encapsulate all the functionality of the generic extension. In practice, it is not uncommon that the Grouped AVP will encapsulate an existing AVP that has previously been defined as mandatory ('M'-bit set), e.g., 3GPP IP Multimedia Subsystems (IMS) Cx/Dx interfaces ([TS29.228] and [TS29.229]).

7. Guidelines for Registrations of Diameter Values

As summarized in Section 3 of this document and further described in Section 1.3 of [RFC6733], there are four main ways to extend Diameter. The process for defining new functionality slightly varies based on the different extensions. This section provides protocol designers with some guidance regarding the definition of values for possible Diameter extensions and the necessary interaction with IANA to register the new functionality.

a. Defining New AVP Values

The specifications defining AVPs and AVP values MUST provide guidance for defining new values and the corresponding policy for adding these values. For example, RFC 5777 [RFC5777] defines the Treatment-Action AVP, which contains a list of valid values corresponding to predefined actions (drop, shape, mark, permit). This set of values can be extended following the Specification Required policy defined in [RFC5226]. As a second example, the Diameter base specification [RFC6733] defines the Result-Code AVP that contains a 32-bit address space used to identity possible errors. According to Section 11.3.2 of [RFC6733], new values can be assigned by IANA via an IETF Review process [RFC5226].

b. Creating New AVPs

Two different types of AVP Codes namespaces can be used to create a new AVP:

* IETF AVP Codes namespace.

* Vendor-specific AVP Codes namespace.

In the latter case, a vendor needs to be first assigned by IANA with a private enterprise number, which can be used within the Vendor-Id field of the vendor-specific AVP. This enterprise number delimits a private namespace in which the vendor is responsible for vendor-specific AVP code value assignment. The absence of a Vendor Id or a Vendor-Id value of zero (0) in the AVP header identifies standard AVPs from the IETF AVP Codes namespace

managed by IANA. The allocation of code values from the IANA-managed namespace is conditioned by an Expert Review of the specification defining the AVPs or an IETF Review if a block of AVPs needs to be assigned. Moreover, the remaining bits of the AVP Flags field of the AVP header are also assigned via Standards Action if the creation of new AVP flags is desired.

c. Creating New Commands

Unlike the AVP Codes namespace, the Command Code namespace is flat, but the range of values is subdivided into three chunks with distinct IANA registration policies:

* A range of standard Command Code values that are allocated via IETF Review;

* A range of vendor-specific Command Code values that are allocated on a first-come, first-served basis; and

* A range of values reserved only for experimental and testing purposes.

As for AVP flags, the remaining bits of the Command Flags field of the Diameter header are also assigned via a Standards Action to create new Command flags if required.

d. Creating New Applications

Similarly, to the Command Code namespace, the Application-Id namespace is flat but divided into two distinct ranges:

* A range of values reserved for standard Application Ids, allocated after Expert Review of the specification defining the standard application.

* A range for values for vendor-specific applications, allocated by IANA on a first-come, first-served basis.

The IANA AAA parameters page can be found at <http://www.iana.org/assignments/aaa-parameters>, and the enterprise number IANA page is available at <http://www.iana.org/assignments/enterprise-numbers>. More details on the policies followed by IANA for namespace management (e.g., first-come, first-served; Expert Review; IETF Review; etc.) can be found in [RFC5226].

NOTE: When the same functionality/extension is used by more than one vendor, it is RECOMMENDED that a standard extension be defined. Moreover, a vendor-specific extension SHOULD be registered to avoid interoperability issues in the same network. With this aim, the registration policy of a vendor-specific extension has been simplified with the publication of [RFC6733], and the namespace reserved for vendor-specific extensions is large enough to avoid exhaustion.

8. Security Considerations

This document provides guidelines and considerations for extending Diameter and Diameter applications. Although such an extension may be related to a security functionality, the document does not explicitly give additional guidance on enhancing Diameter with respect to security. However, as a general guideline, it is recommended that any Diameter extension SHOULD NOT break the security concept given in [RFC6733]. In particular, it is reiterated here that any command defined or reused in a new Diameter application SHOULD be secured by using TLS [RFC5246] or DTLS/SCTP [RFC6083] and MUST NOT be used without one of the following: TLS, DTLS, or IPsec [RFC4301]. When defining a new Diameter extension, any possible impact of the existing security principles described in [RFC6733] MUST be carefully appraised and documented in the Diameter application specification.

9. References

9.1. Normative References

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

[RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012, <http://www.rfc-editor.org/info/rfc6733>.

9.2. Informative References

[Q.3303.3] International Telecommunications Union, "Resource control protocol No. 3: Protocols at the Rw interface between the policy decision physical entity (PD-PE) and a policy enforcement physical entity (PE-PE): Diameter profile version 3", ITU-T Recommendation Q.3303.3, August 2008.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998, <http://xml.resource.org/public/rfc/info/rfc2409>.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003, <http://www.rfc-editor.org/info/rfc3588>.

[RFC4005] Calhoun, P., Zorn, G., Spence, D., and D. Mitton, "Diameter Network Access Server Application", RFC 4005, August 2005, <http://www.rfc-editor.org/info/rfc4005>.

[RFC4072] Eronen, P., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, August 2005, <http://www.rfc-editor.org/info/rfc4072>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005, <http://www.rfc-editor.org/info/rfc4301>.

[RFC4740] Garcia-Martin, M., Belinchon, M., Pallares-Lopez, M., Canales-Valenzuela, C., and K. Tammi, "Diameter Session Initiation Protocol (SIP) Application", RFC 4740, November 2006, <http://www.rfc-editor.org/info/rfc4740>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5447] Korhonen, J., Bournelle, J., Tschofenig, H., Perkins, C., and K. Chowdhury, "Diameter Mobile IPv6: Support for Network Access Server to Diameter Server Interaction", RFC 5447, February 2009, <http://www.rfc-editor.org/info/rfc5447>.

[RFC5777] Korhonen, J., Tschofenig, H., Arumaithurai, M., Jones, M., and A. Lior, "Traffic Classification and Quality of Service (QoS) Attributes for Diameter", RFC 5777, February 2010, <http://www.rfc-editor.org/info/rfc5777>.

[RFC6083] Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC 6083, January 2011, <http://www.rfc-editor.org/info/rfc6083>.

[RFC6735] Carlberg, K. and T. Taylor, "Diameter Priority Attribute-Value Pairs", RFC 6735, October 2012, <http://www.rfc-editor.org/info/rfc6735>.

[RFC7075] Tsou, T., Hao, R., and T. Taylor, "Realm-Based Redirection In Diameter", RFC 7075, November 2013, <http://www.rfc-editor.org/info/rfc7075>.

[RFC7155] Zorn, G., "Diameter Network Access Server Application", RFC 7155, April 2014, <http://www.rfc-editor.org/info/rfc7155>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[TS29.228] 3rd Generation Partnership Project, "Technical Specification Group Core Network and Terminals; IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents", 3GPP TS 29.228, September 2014, <http://www.3gpp.org/ftp/Specs/html-info/29228.htm>.

[TS29.229] 3rd Generation Partnership Project, "Technical Specification Group Core Network and Terminals; Cx and Dx interfaces based on the Diameter protocol; Protocol details", 3GPP TS 29.229, September 2014, <http://www.3gpp.org/ftp/Specs/html-info/29229.htm>.

[TS29.328] 3rd Generation Partnership Project, "Technical Specification Group Core Network and Terminals; IP Multimedia (IM) Subsystem Sh interface; Signalling flows and message contents", 3GPP TS 29.328, September 2014, <http://www.3gpp.org/ftp/Specs/html-info/29328.htm>.

[TS29.329] 3rd Generation Partnership Project, "Technical Specification Group Core Network and Terminals; Sh Interface based on the Diameter protocol; Protocol details", 3GPP TS 29.329, September 2014, <http://www.3gpp.org/ftp/Specs/html-info/29329.htm>.

Contributors

The content of this document was influenced by a design team created to revisit the Diameter extensibility rules. The team was formed in February 2008 and finished its work in June 2008. In addition to those individuals listed in the Authors' Addresses section, the design team members were:

- Avi Lior
- Glen Zorn
- Jari Arkko
- Jouni Korhonen
- Mark Jones
- Tolga Asveren
- Glenn McGregor
- Dave Frascone

We would like to thank Tolga Asveren, Glenn McGregor, and John Loughney for their contributions as coauthors to earlier versions of this document.

Acknowledgments

We greatly appreciate the insight provided by Diameter implementors who have highlighted the issues and concerns being addressed by this document. The authors would also like to thank Jean Mahoney, Ben Campbell, Sebastien Decugis, and Benoit Claise for their invaluable, detailed reviews and comments on this document.

Authors' Addresses

Lionel Morand (editor)
Orange Labs
38/40 rue du General Leclerc
Issy-Les-Moulineaux Cedex 9 92794
France

Phone: +33145296257
EMail: lionel.morand@orange.com

Victor Fajardo
Fluke Networks

EMail: vf0213@gmail.com

Hannes Tschofenig
Hall in Tirol 6060
Austria

EMail: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at