DOF Tunnel Protocol

DOF Protocol Specification

Technical Steering Committee

DOF Tunnel Protocol: DOF Protocol Specification

Technical Steering Committee

Document Information

Document Tracking Number	dof-2015-1
Document Version	7.0.1, 12 January 2018
Build Version	7.0.1.8
Source Information	2018-01-12, "." at commit c4a21c8bf741a32d4fe20776861d03baab74639a on branch "master" of repository core-specifications
External Reference	DOF Tunnel Protocol, OpenDOF TSC, [dof-2015-1] (7.0.1, 12 January 2018)

This document is managed by the Technical Steering Committee of the OpenDOF Project, Inc.. Contact the committee using the following information.

Contact Information

Post Mail	OpenDOF Project, Inc. Technical Steering Committee 3855 SW 153rd Drive Beaverton, OR 97003
Telephone	1-503-619-4114
Fax	1-503-644-6708
Email	tsc@member.opendof.org [mailto:tsc@member.opendof.org]

Copyright

Copyright © 2008-2018 Panasonic Corporation, Bryant Eastham

Patents

Contributors to the OpenDOF Project make a non-enforcement pledge related to any patent required to implement the protocols described in this document. For more information see https://opendof.org/pledge.

License

See https://opendof.org/license-specifications.

Table of Contents

1. About This Document 1
    1.1. Audience 1
    1.2. Authority 1
    1.3. PDU Definitions 1
        1.3.1. Context 1
        1.3.2. Generalization 2
        1.3.3. Qualifications 2
        1.3.4. Fields 4
    1.4. Specifications, Notes and Warnings 5
    1.5. Printing This Document 6
    1.6. Comparing Documents 6
        1.6.1. Text Comparison 6
        1.6.2. Graphical Comparison 6
2. Specification 9
    2.1. Tunnel Protocol PDU 9
    2.2. Version Details 9
        2.2.1. Version 0 9
        2.2.2. Version 1 10
    2.3. Negotiation 11
    2.4. Tunnel Lifecycle 11
    2.5. Tunnel Opening 11
    2.6. Open Behavior 11
    2.7. Closed 12
    2.8. Version 0 and 1 PDUs 12
        2.8.1. Open PDU 12
        2.8.2. Opened PDU 13
        2.8.3. Send PDU 13
        2.8.4. Close PDU 14
        2.8.5. Closed PDU 15
3. External PDU Reference 17
    3.1. DOF Common Types, OpenDOF TSC, [dof-2009-1] (7.0.1, 12 January 2018) 17
    3.2. DOF Security Specification, OpenDOF TSC, [dof-2008-16] (7.0.1, 12 January 2018) 18

Specifications and PDUs

General Tunnel Packet `dof-2015-1-pdu-1` 9
Tunnel Version 0 Packet `dof-2015-1-pdu-2` 10
Tunnel Version 1 Packet `dof-2015-1-pdu-3` 10
Open `dof-2015-1-pdu-4` 12
Opened `dof-2015-1-pdu-5` 13
Send `dof-2015-1-pdu-6` 13
Close `dof-2015-1-pdu-7` 14
Closed `dof-2015-1-pdu-8` 15
Compressed Unsigned 16-Bit `dof-2009-1-pdu-1` 17
Compressed Unsigned 24-Bit `dof-2009-1-pdu-2` 17
Identity Resolution `dof-2008-16-pdu-3` 18

1. About This Document

This document contains the protocol specification of the Negotiated Tunnel Protocol. This is an application bridging protocol. This document covers protocol versions 0 and 1.

1.1. Audience

The primary audience of this document is people implementing libraries that use DOF protocols. Others that are concerned about how DOF protocols work on a network at a very low level can also benefit from this document.

Readers should be familiar with technical protocol documentation. This document is similar to an 'RFC' for DOF protocols, and familiarity with the language used in these types of documents is helpful.

This document is *not* required reading for those who need to use existing DOF libraries, although system designers may benefit from an understanding of this information. In particular, the information related to security and connections can help system designers to better understand how DOF systems work.

1.2. Authority

This document, in its English form, is the authoritative document for DOF protocols. Due to the detailed nature of protocol design, it may be difficult for a reader to implement these protocols correctly based on a translated document. When questions arise, the English version is authoritative.

All implementations that claim to be DOF compliant must satisfy the requirements set forth in this document.

This document is managed by the Technical Steering Committee of the OpenDOF Project, Inc., referred to as 'ODP-TSC'. The inside cover of this document contains contact information for the Technical Steering Committee.

The web site for the Technical Steering Committee is https://opendof.org/tsc.

1.3. PDU Definitions

This document deals with the transmission of information on a network. In order to describe this, diagrams show different fields, their network order, and positioning.

All DOF protocol documentation uses a similar diagramming style, discussed in this section. The word 'PDU', which stands for 'Protocol Data Unit', indicates these diagrams.

1.3.1. Context

Each PDU represents information sent on the network in some context. This usually means that there is information sent 'surrounding' the PDU itself. This information provides the 'context' for the PDU.

For example, the context for the DOF Protocol Stack itself is a network transport. This could be UDP/IP, TCP/IP, or any other transport. The transport will require certain headers and trailers, but they are not shown in each DOF Protocol Stack PDU. In this example the transport provide the context of the DOF Protocol Stack PDU.

In a similar way, the DOF Protocol Stack defines the context for the other DOF protocols. The Object Access Protocol, for example, can use the DOF Protocol Stack for its context. Just as the transport headers are not shown for the DOF Protocol Stack, the DOF Protocol Stack headers are not shown for the Object Access Protocol.

Understanding the context requires understanding the relative position of the different layers in the protocol stack. Each PDU defines the ordering of all of its parts, and may indicate that it is itself an instance of some other general PDU format. This means that it is always possible to start at a high-level PDU definition and understand it completely, but not always possible to identify everywhere that a particular PDU may be used (as a part of some other higher-level PDU).

Each protocol specification defines its specific contexts. The context definition includes a description of the underlying stack layer (or the containing layer), and defines the information that must be available from that layer (for reception) or passed to that layer (for transmission).

As an example, the context of the DOF Network Protocol is a transport. The transport has a requirement to provide the addresses of nodes, and so part of the context for the transport is the sender address (for reception) or the target address (for transmission). Any transport that will work with the DOF Protocol Stack must meet the context requirements.

1.3.2. Generalization

Many times a particular PDU will be a specific type of a more general PDU. In these cases, the PDU will indicate that it is an 'instance' of the more general type.

1.3.3. Qualifications

Each PDU will have qualifications placed on its use. The general qualifications include security and transport requirements. The following terms identify these qualifications. Others may be present as well.

1.3.3.1. Transport Qualifications

There are three major transport categories for DOF protocols (see the *DOF Protocol Specification*, Transport Requirements). These are None, Lossy, and Lossless. There are also three addressing types: Unicast, Multicast, and Broadcast. PDUs indicate the ability to use each of these categories as follows:

```
Session: Combinations of None, Lossy (2-node, n-node), or Lossless.
Addressing: Combinations of Unicast, Multicast, or Broadcast.
```

1.3.3.2. Security Qualifications

Each PDU categorizes security qualifications according to the major aspects of security (see the *DOF Security Specification*, Overview). These are Encryption, Data Integrity, Authentication, and Access Control. Encryption prevents viewing of the PDU, Data Integrity ensures that others do not modify the PDU, Authentication indicates knowledge of the nodes involved in the communication, and Access Control restricts those able to use the PDU.

Data Integrity is required according to the DOF Protocol Stack specification for all secure packets. Authentication is required whenever permissions are involved, or when it is important to know something about the relationship between the sender and receiver. Encryption is required if the contents of the packet contain information that must not be visible to potential attackers.

In addition, there is the case of unsecured communication. PDUs that do not require security will list 'Unsecured' as a possible choice. Finally, certain PDUs are required to be Unsecured, and are identified as such.

Each application PDU must specify its security requirements. If Access Control is required, the PDU will indicate the permissions required for the command and for any response. However, it is typical for all of the PDUs of a given protocol to share the same requirements, and so the protocol specification may define its security qualifications in a single section. In this case, the individual PDUs should refer back to this section to avoid confusion.

It is also typical for applications to require the security of the session. A PDU indicates this as 'session security', and is its own security qualification. If the PDU allows the 'None' session type then this implies that Unsecured is allowed, as session type 'None' cannot be secure.

For PDUs that refer to session security or a common security section, one of the following formats is used:

```
Security: See the section 'Security Qualifications'.
```

or

```
Security: Session.
```

The PDU uses the following identifications to specify the security qualifications for a PDU:

```
Unsecured: (description).
Encryption: (description).
Message authentication: (description).

Permission (command): (description).
Permission (response): (description).
```

For each of these, the (description) varies. There are many different combinations of values, as described here.

Unsecured can have the values 'Required', 'Allowed', 'Not Allowed'.

Encryption can have the values 'Required', 'Allowed', 'Not Allowed'.

Message authentication can have the values 'Required', 'Allowed', 'Not Allowed'.

The specific PDU determines the permissions required (for both command and response).

There are combinations that are not valid. The following table defines all allowed combinations if a combination does not appear in the table then it is invalid.

In general, security theory links Encryption and Data Integrity. If one is 'Required' then the other must at least be 'Allowed'. Encryption and Data Integrity are the opposite of Unsecured. If either is 'Required' then unsecured must be 'Not Allowed'. Again, note that since data integrity is a common requirement for all secure packets the PDUs do not list it separately.

Unsecured / Encryption / Message Authentication	**Description**
Required / Not Allowed / Not Allowed	Indicates the PDU must be unsecured.
Allowed / Allowed / Allowed	Indicates there are no security requirements.
Not Allowed / Allowed / Allowed	Indicates the PDU must be either encrypted or authenticated. This is the lowest level of security possible for a PDU.

Unsecured / Encryption / Message Authentication	Description
Not Allowed / Required / Allowed	Indicates that encryption is required, and data integrity is optional.
Not Allowed / Allowed / Required	Indicates that encrypted is optional, but authentication is required.
Not Allowed / Required / Required	Indicates both encryption and authentication are required.

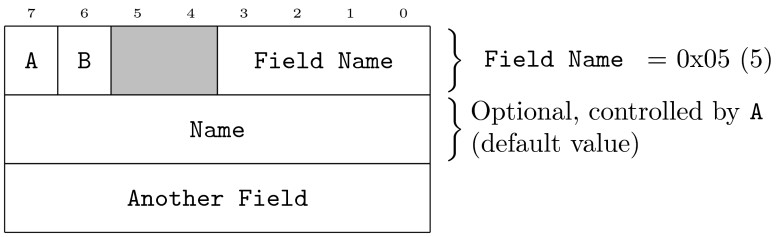
1.3.4. Fields

Each PDU shows a container, represented as a sequence of fields. The table fully describes each field.

The following is an example of a PDU description.

Example PDU`example-pdu-1`

PDU Context



`A`	**One bit.** Controls the presence of `Name`.
`B`	**One bit.** Field description.
`Field Name`	**Type and size description.** Field description.
`Name`	**Type and size description. Optional, controlled by** `A`. Field description.
`Another Field`	**Type and size description.** Field description.

The container represents the PDU. The diagram shows Fields first to last, left to right and top to bottom. This means that the first fields appear on the wire before the last fields. This is a general principle: things on the top and toward the left in the diagrams appear on the wire before things on the right and bottom. Note that the transport is free to use whatever ordering it must, and memory layout may different from the PDU definition. However, the transfer of a PDU from a program, over an arbitrary transport and into another program must maintain the perceived ordering between the two programs. PDU diagrams are always shown MSB first (most significant byte) and msb first (most significant bit) unless otherwise indicated.

Each field identifies its contents and provides information necessary to understand its use. In the example above, the diagram shows the first six fields of the container. Each field is named, with the exception of reserved fields which are indicated as being grayed out. Field names use `monospace text` similar to that used in program listing. The third field shown is indicated as reserved. Reserved bits have special requirements described in the *DOF Protocol Specification*.

Several different field cases are typical, described in the following sections.

1.3.4.1. Typical Field

A typical field defines its name, its type, and its description. As shown, this field is not optional, and so it must appear in the position indicated by the diagram. The PDU uses `Field Name` throughout the related documentation to refer to the field. The type of the field is a reference to some other defined PDU. Otherwise, the PDU indicates the size of the field and any defined structure.

1.3.4.2. Optional Field

An optional field is similar to the typical case, but is indicated as optional in a note to the right. Each optional field will indicate what controls whether or not it is included as well as the default value that the field takes on when it is absent.

1.3.4.3. Fixed Values

Fields may have required values based on the specific PDU. This is common in the case of a PDU being an instance of another PDU, where the specific instance requires certain field values. In this case, it is not valid for the PDU to contain values other than that specified, if so the PDU is invalid.

The required value is shown to the right of the field as a comment.

1.3.4.4. Bit Fields

It is common for PDUs to leverage a single byte to store multiple fields (bit fields). The packing of multiple fields into a single byte is indicated as shown in the earlier example. This packing of bits may also leave space that can be indicated as reserved.

The PDU indicates reserved bits visually although the PDU does not individual name these fields.

Note that bit fields may specify default values just as other fields. However, single bit fields do not show their hexadecimal representation. In addition, bits without names that have required values may indicate their values in the place of the name.

1.4. Specifications, Notes and Warnings

This is a technical specification document. It is critical to understand how the document indicates specification items. There are also annotations for notes (implementation notes or other information that is worthy of extra notice) and warnings (pitfalls or extremely critical information). In addition, there are implementation notes, which point out optimizations that are not obvious.

Each type of callout ends either when another callout begins or when the text goes back to the normal indentation.



This is a note. The text is inset and there is a header to indicate the beginning of the text.



This is an implementation note. The text is inset and there is a header to indicate the beginning of the text.



This is a warning. The text is bold inset and there is a different header to indicate the beginning of the text.



This is an example`example-spec-1`

This is the text.

Following the summary is descriptive information about the specification item. Each specification item has a unique number assigned.

1.5. Printing This Document

The format of this document is for two-sided printing on US Letter paper.

1.6. Comparing Documents

The easiest way to compare versions of this document is to compare the corresponding source. The source format is AsciiDoctor, which is a markup-style text format. This format can be compared using any text comparison tool, including those that work with source control documents. To facilitate determining the source for each document there is information about the commit, branch, and repository along with the document version information. In addition the following guidelines are followed to make things easy to compare:

1. The input files put each sentence on a separate line, with no hard line-wrapping.

2. To the extent possible, macros and templates are used to determine formatting and structure.

3. Graphics and diagrams also use text formats.

If all you have available is a compiled document (PDF or HTML) then there are tools available to compare them. This section summarizes some of these methods with their pros and cons.

1.6.1. Text Comparison

There are several free tools that will do comparisons of the text of a PDF. One of these is the xdocdiff plugin for WinMerge. The plugin extracts the text from the PDF (including some information like page numbers), and WinMerge then compares the text.

WinMerge is available from http://winmerge.org. The xdocdiff plugin is available from http://freemind.s57.xrea.com/xdocdiffPlugin/en/index.html.

1.6.2. Graphical Comparison

In order to compare formatting, or to see the changes in the context of the document, a graphical compare is required. There are several free tools, but each has limitations.

1. diff-pdf, available from https://github.com/vslavik/diff-pdf, shows an overlay of the matching pages in different colors.

2. DiffPDF, available fromhttp://www.qtrac.eu/diffpdf.html, shows side-by-side red-lined differences of matching pages.

Both of these tools suffer from their page-by-page comparison methods. This means that when context shifts because of additions or removals that they start sensing changes at each page boundary. DiffPDF has the ability to select comparison ranges, allowing the program to synchronize at the beginning of each section, but this requires manual configuration and can be difficult for large documents.

There are also commercial tools that do an excellent job at comparing PDF documents. The first is Adobe Acrobat (http://www.adobe.com/products/acrobat.html), which in later releases has a 'Compare Document' feature. The result of the comparison is an annotated PDF, and hovering the mouse over the change will show the details.

Another commercial tool is PDF Content Comparer (http://www.inetsoftware.de/products/pdf-content-comparer) from i-net Software. This tool does side-by-side comparison, but has an intuitive auto-synchronize that keeps similar content lined up even when additions or deletions have occurred.

2. Specification

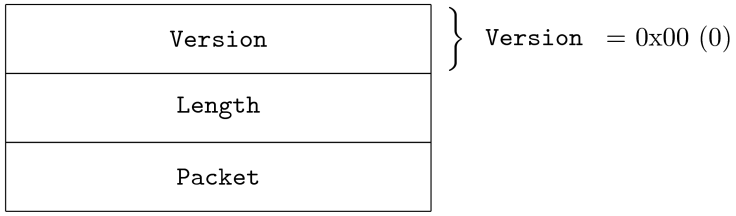
The purpose of this protocol is to tunnel application traffic from a client to a server. Each version of the protocol has different tunneling characteristics and uses. The following table presents general information on each version.

Version	Description
0	DOF over SSL with Security Bridging.
1	DOF over TCP, unsecured.

2.1. Tunnel Protocol PDU

PDUs in this protocol follow the same general format, described here, independent of version.

⇄ **General Tunnel Packet** `dof-2015-1-pdu-1`



`Version` **One byte.**

`Length` **Two bytes, MSB.** The length of the **Packet** data, which may be zero.

`Packet` **Variable length. Optional, controlled by** `Length`. Present if **Length** is greater than zero.

Note that negotiation packets have a **Length** of zero (0), and so all packets that can be sent as the first non-negotiation packet must have a non-zero length in order to distinguish version assertion (described later).

2.2. Version Details

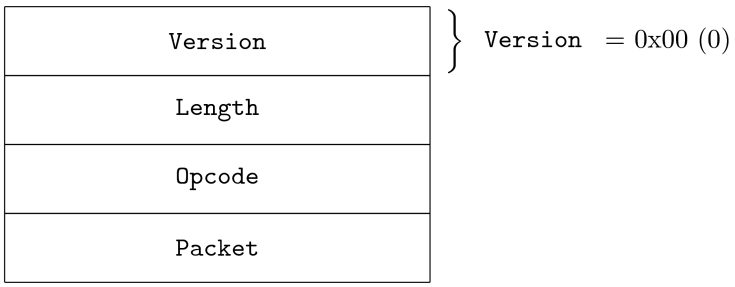
2.2.1. Version 0

This version of the protocol allows a client to use DOF datagram protocols and offload security enforcement to a client and requires a transport that provides privacy and server authentication.

Note that it is critical that the client trust the server *before using this protocol*. This protocol immediately sends plaintext credentials to the secure server, and so the client *must* establish trust of the server *before* sending these plaintext credentials.

PDUs in this protocol follow the same general format, described here, independent of version.

⇄ Tunnel Version 0 Packet `dof-2015-1-pdu-2`



`Version` **One byte.**

`Length` **Two bytes, MSB.** The length of the **Opcode** and **Packet** data, which must be greater than zero.

`Opcode` **One byte.** Specified by the PDU.

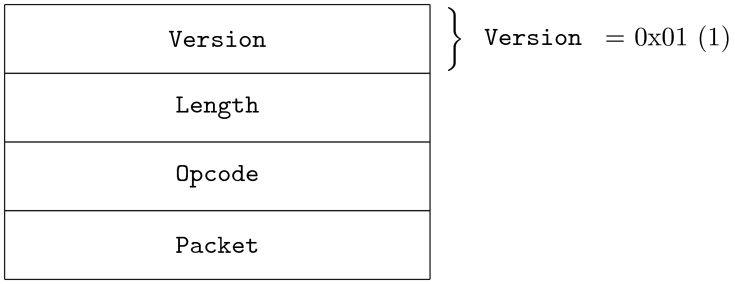
`Packet` **Variable length. Optional, controlled by** `Length`. Present if **Length** is greater than one.

2.2.2. Version 1

This version of the protocol allows a client to use EMIT DO datagram protocols with no security. It is possible to use a secure transport with this protocol version, but it is not required.

PDUs in this protocol follow the same general format, described here, independent of version.

⇄ Tunnel Version 1 Packet `dof-2015-1-pdu-3`



`Version` **One byte.**

`Length` **Two bytes, MSB.** The length of the **Opcode** and **Packet** data, which must be greater than zero.

`Opcode` **One byte.** Specified by the PDU.

`Packet` **Variable length. Optional, controlled by** `Length`. Present if **Length** is greater than one.

2.3. Negotiation

The first step in negotiation is to open an appropriate connection to an established server. There are two ports assigned to this protocol. For unsecure communication port **8567** is used. For secure transports port **3568** is used.

The version of the protocol used on the connection is negotiated upon opening the connection and begins with the client. The client sends a zero-length packet (meaning the **Version** byte followed by a two-byte zero **Length** field). If the protocol identifier is acceptable to the server then the server responds with the same PDU. If the server cannot understand the version requested by the client and a different version of the same protocol is supported, then the response PDU will include that **Version** and if the client approves it will echo that PDU. This proceeds, with the client and server exchanging roles, until they converge on the same **Version**. This negotiation must proceed from highest **Version** to lowest. The client and server must converge within 10 seconds or the connection must be terminated.

Note that the set of versions allowed on a particular connection may depend on the security of the transport. For example, version 0 requires a secure transport, and attempts to negotiate version 0 on port 8567 will fail.

Alternatively, a client may bypass negotiation and immediately begin using a protocol identifier. This case is identified by the initial packet having a **Length** greater than zero. If the server cannot accept the PDU then it will close the connection.

2.4. Tunnel Lifecycle

The tunnel goes through the following stages:

1. Initial – not open.
2. Transport Opened – the transport connection is established, and the client establishes trust of the server if required.
3. Negotiation – the client and server agree on protocol versions.
4. Tunnel Opening – the client sends a credential and the server accepts (or rejects) it.
5. Open – the client and server exchange packets.
6. Closed – the client or server close the tunnel and transport connection.

2.5. Tunnel Opening

After opening the transport and negotiating the protocol version, the client has 30 seconds to establish a tunnel. Each version determines the PDUs necessary to open the tunnel.

2.6. Open Behavior

Both the client and server send packets, and they may do so at any time and with no acknowledgement.

Note that some systems may require periodic traffic in order to keep the session alive. In these cases the side of the system that knows it has this constraint can use either a PDU with a **Length** equal to zero, or a version-specific PDU.

2.7. Closed

The tunnel can be closed by either the server or client terminating the connection (not preferred) or sending whatever version-specific PDUs are defined.

2.8. Version 0 and 1 PDUs

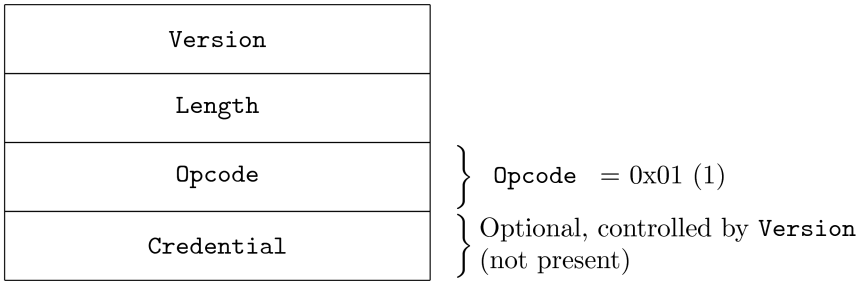
Version 0 and version 1 are closely related, and so are described together. The only difference relates to the passing of a credential in the Open PDU, which is required for version 0 and prohibited for version 1.

2.8.1. Open PDU

This section describes the Open PDU, which is the first PDU sent by the client after negotiation.

⇄ **Open** `dof-2015-1-pdu-4`

Instance of Tunnel Version 0 Packet`dof-2015-1-pdu-2`
or
Instance of Tunnel Version 1 Packet`dof-2015-1-pdu-3`



`Version` **One byte.**

`Length` **Two bytes, MSB.** The length of the **Opcode** and **Packet** data, which must be greater than zero.

`Opcode` **One byte.** Specified by the PDU.

`Credential` **Instance of** Identity Resolution`dof-2008-16-pdu-3`.**Optional, controlled by** `Version`. The credential the client wishes to use for authentication and access control. Defined as the EMIT DO Credential format, stage 255 (storage). See the *EMIT DO Security Specification* and the credential specification for more information. This field is required for version 0 and prohibited for version 1.

The Open PDU establishes the tunnel and for version 0 encapsulates authentication information. The authentication information includes a domain, identity, and secret information. The credential bytes can be obtained by calling DOFCredentials.getBytes() call, and the DOFCredentials.create() method can be used on those bytes to recreate the original credential.

ⓘ This PDU for version 0 exposes security credentials over the wire. The sender must ensure that the transport connection is encrypted, that the receiver (server) is trusted, and that there can be no

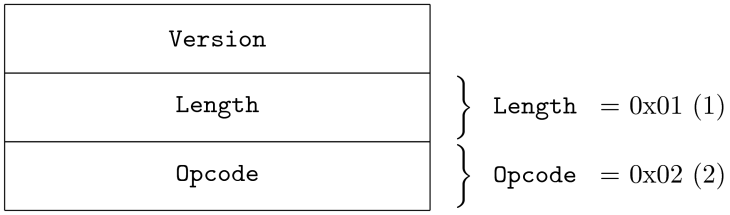
man-in-the-middle attacks against the transport. Improper use of this PDU will result in the loss of secure information and the ability of an attacker to utilize the credentials for other connections, impersonating the sender.*

2.8.2. Opened PDU

This PDU is sent by the server if the tunnel is successfully opened.

⇄ **Opened** `dof-2015-1-pdu-5`

Instance of Tunnel Version 0 Packet`dof-2015-1-pdu-2`
or
Instance of Tunnel Version 1 Packet`dof-2015-1-pdu-3`



`Version` **One byte.**

`Length` **Two bytes, MSB.** The length of the **Opcode** and **Packet** data, which must be greater than zero.

`Opcode` **One byte.** Specified by the PDU.

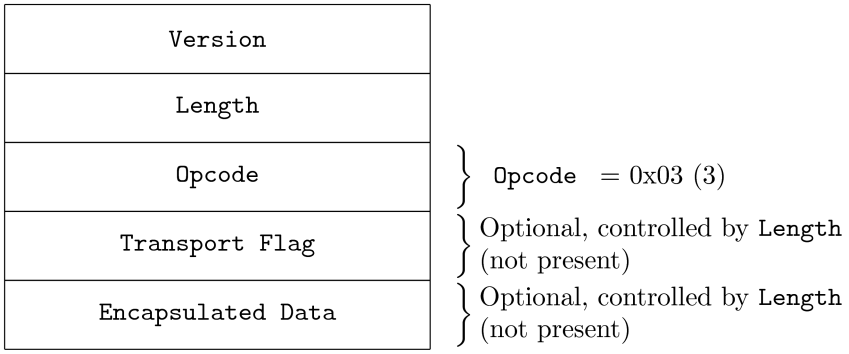
The Opened PDU is the successful response to the Open PDU, and is sent by the server. If the tunnel is not successful (the credential is invalid), then the tunnel is closed by the server.

2.8.3. Send PDU

The Send PDU is used by the client and server to exchange PDUs after the tunnel is opened.

⇄ **Send** `dof-2015-1-pdu-6`

Instance of Tunnel Version 0 Packet`dof-2015-1-pdu-2`
or
Instance of Tunnel Version 1 Packet`dof-2015-1-pdu-3`



`Version` **One byte.**

`Length` **Two bytes, MSB.** The length of the **Opcode** and **Packet** data, which must be greater than zero.

`Opcode` **One byte.** Specified by the PDU.

`Transport Flag` **One byte. Optional, controlled by** `Length`. A flag, zero (0) indicating a unicast packet and one (1) indicating a multicast packet. Other values are illegal and result in the session being terminated. The field is present whenever **Length** is greater than one (1).

`Encapsulated Data` **Variable length. Optional, controlled by** `Length`. The encapsulated DOF packet. This is a complete PDU as defined in the *EMIT DO Protocol Specification*. The field is present whenever **Length** is greater than two (2).

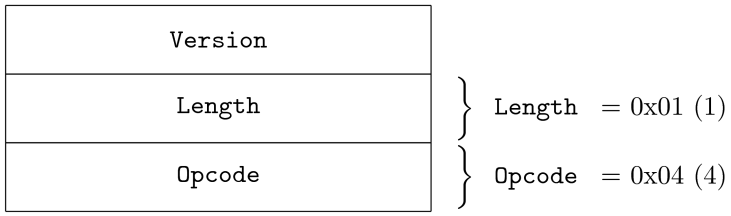
The Send PDU exchanges encapsulated packets between the client and the server. Note that a Send PDU with a **Length** of one (1) can be used as a keep-alive.

2.8.4. Close PDU

This PDU is used by the client or server to initiate a shutdown of the tunnel. The transport connection is also shut down when the tunnel terminates.

⇄ **Close** `dof-2015-1-pdu-7`

Instance of Tunnel Version 0 Packet`dof-2015-1-pdu-2`
or
Instance of Tunnel Version 1 Packet`dof-2015-1-pdu-3`



`Version` **One byte.**

`Length` **Two bytes, MSB.** The length of the **Opcode** and **Packet** data, which must be greater than zero.

`Opcode` **One byte.** Specified by the PDU.

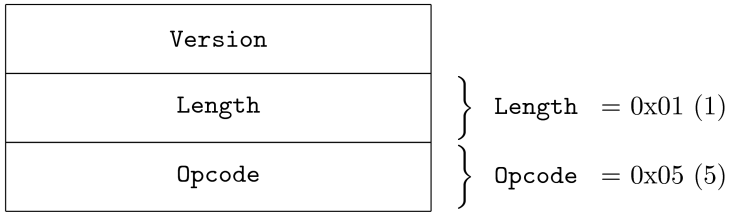
The receiving node should echo a Closed PDU and then terminate the transport connection.

2.8.5. Closed PDU

This PDU is used by the client or server to acknowledge the shutdown of the tunnel. The transport connection is also shut down when the tunnel terminates.

⇄ **Closed** `dof-2015-1-pdu-8`

Instance of Tunnel Version 0 Packet`dof-2015-1-pdu-2`
or
Instance of Tunnel Version 1 Packet`dof-2015-1-pdu-3`



`Version` **One byte.**

`Length` **Two bytes, MSB.** The length of the **Opcode** and **Packet** data, which must be greater than zero.

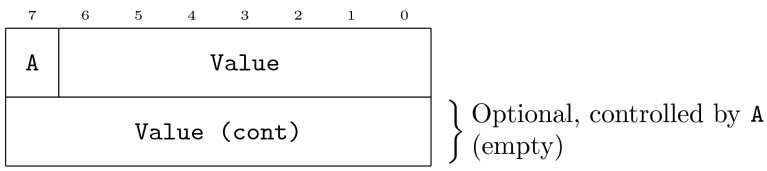
`Opcode` **One byte.** Specified by the PDU.

The receiving node may wait for this packet, but it is optional and may not be received. The transport connection should be shut down after a small time period.

3. External PDU Reference

3.1. DOF Common Types, OpenDOF TSC, [dof-2009-1] (7.0.1, 12 January 2018)

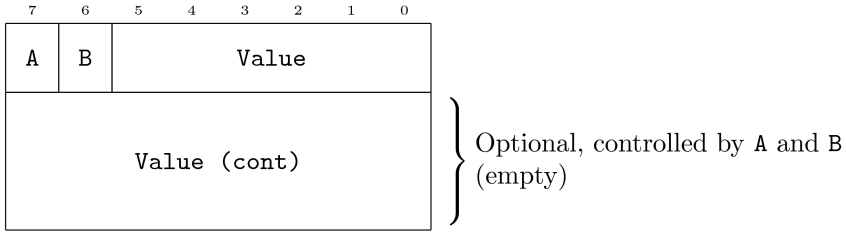
Compressed Unsigned 16-Bit `dof-2009-1-pdu-1`



`A` **One bit.** Controls the size of `Value`.

`Value` **Either seven or fifteen bits, MSB, controlled by A.** If the `A` field is zero then the `Value` is contained in a single byte. If the `A` field is one then the `Value` is passed in fifteen bits with the least significant bits passed in a second byte.

Compressed Unsigned 24-Bit `dof-2009-1-pdu-2`



`A` **One bit.** Controls the size of `Value` and the presence of the `B` field.

`B` **One bit.** Controls the size of `Value` or becomes part of the value.

`Value` **Either seven, fourteen, or twenty-two bits, MSB, controlled by A and B.** If `A` is zero the `Value` is seven bits and one byte is used. In this case, the `B` field is part of the `Value`. If `A` is one then the `B` field determines the size of the `Value` rather than being part of it. If `B` is zero then fourteen bits (two bytes) are used, and if `B` is one then twenty-two bits (three bytes) are used.

3.2. DOF Security Specification, OpenDOF TSC, [dof-2008-16] (7.0.1, 12 January 2018)

⇄ **Identity Resolution** `dof-2008-16-pdu-3`

Credential Type
Stage
Length
Public Data

`Credential Type` **Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.

`Stage` **One byte.** The stage of the request. This value must be greater than zero and less than 255.

`Length` **Instance of** Compressed Unsigned 24-Bit`dof-2009-1-pdu-2`.Controls length of **Public Data**.

`Public Data` **Variable length. Optional, controlled by** `Length`. The public data associated with the **Credential Type** and **Stage**.