Shared Key Identification Specification

DOF Protocol Specification

Technical Steering Committee

Shared Key Identification Specification: DOF Protocol Specification

Technical Steering Committee

Document Information

Document Tracking Number	dof-2008-14
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	Shared Key Identification Specification, OpenDOF TSC, [dof-2008-14] (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. Definitions and PDUs 9
    2.2. Storage 11
    2.3. Persistence 11
    2.4. Shared-key Cryptography 12
3. External PDU Reference 13
    3.1. DOF Common Types, OpenDOF TSC, [dof-2009-1] (7.0.1, 12 January 2018) 13
    3.2. DOF Protocol Specification, OpenDOF TSC, [dof-2008-1] (7.0.1, 12 January 2018) 15
    3.3. DOF Security Specification, OpenDOF TSC, [dof-2008-16] (7.0.1, 12 January 2018) 16

Specifications and PDUs

Shared-key Identification `dof-2008-14-pdu-1` 9
Shared-key Resolution (request, stage 1) `dof-2008-14-pdu-2` 9
Shared-key Resolution (response, stage 1) `dof-2008-14-pdu-3` 10
Shared-key Persistence `dof-2008-14-pdu-4` 11
Compressed Unsigned 16-Bit `dof-2009-1-pdu-1` 13
Compressed Unsigned 24-Bit `dof-2009-1-pdu-2` 13
Compressed Unsigned 32-Bit `dof-2009-1-pdu-3` 14
Object Identifier (OID) `dof-2009-1-pdu-4` 14
Object Identifier Attribute `dof-2009-1-pdu-5` 15
Attribute/Value Pair `dof-2008-1-pdu-16` 15
Security Node Identifier `dof-2008-16-pdu-1` 16
Base Credential Format `dof-2008-16-pdu-2` 16
Identity Resolution `dof-2008-16-pdu-3` 17
Identity Persistence `dof-2008-16-pdu-4` 17
Security Domain `dof-2008-16-pdu-5` 17

1. About This Document

This document presents technical specifications for using shared-key identification in DOF systems.

General requirements that apply to all identification are presented in the *DOF Security Specification*.

This information is highly technical. This document serves as a specification and reference against which someone could implement networked software that interoperated with DOF products without using existing libraries.

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.

Unsecured / Encryption / Message Authentication	Description
Not Allowed / Allowed / Allowed	Indicates the PDU must be either encrypted or authenticated. This is the lowest level of security possible for a PDU.
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.

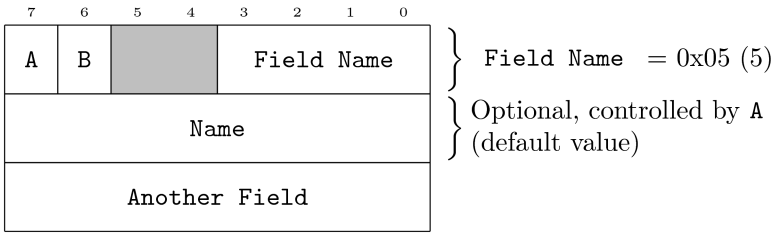
1.3.4. Fields

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

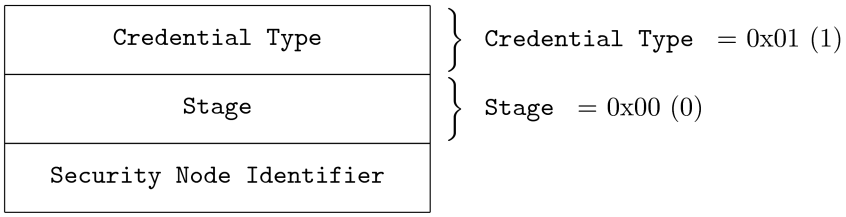
In this method the user is identified with an Object Identifier. Resolution can be used during domain discovery, and since there is a single stage the discovery will be complete without further exchanges.

Note that shared-key identification defines a resolution message (stage 1), even though it is rarely required. The reason is that domain discovery uses the resolution process. The resolution process should not be used except during domain discovery.

2.1. Definitions and PDUs

⇄ Shared-key Identification `dof-2008-14-pdu-1`

Instance of Base Credential Format`dof-2008-16-pdu-2`



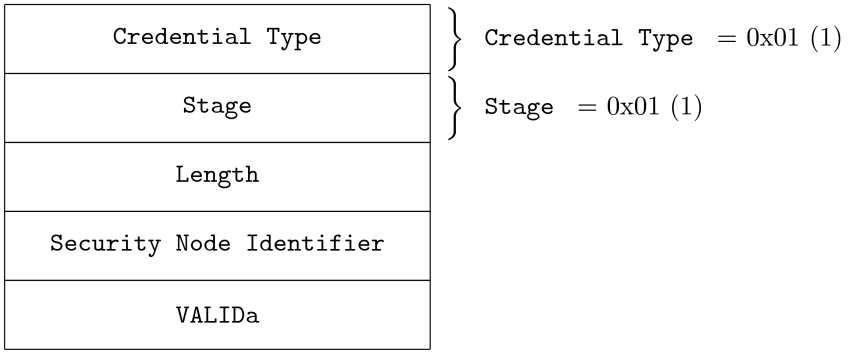
`Credential Type` **Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.

`Stage` **One byte.** The stage of the request. The indicated value is required. This value identifies the PDU as not including a length. The length of the **Security Node Identifier** is contained inside itself, any additional length would be redundant.

`Security Node Identifier` **Instance of** Security Node Identifier`dof-2008-16-pdu-1`.Contains the identity as known by the authentication server.

⇄ Shared-key Resolution (request, stage 1) `dof-2008-14-pdu-2`

Instance of Identity Resolution`dof-2008-16-pdu-3`



`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 the remaining fields.

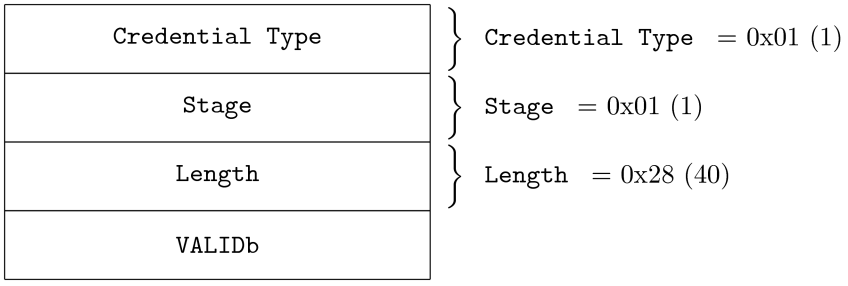
`Security Node Identifier` **Instance of** Security Node Identifier`dof-2008-16-pdu-1`.Contains the identity as known by the authentication server.

`VALIDa` **Forty bytes.** The value of VALID_A as discussed in the specification.

The server must ensure that the VALID_A matches its own calculation before proceeding. If it does, then it sends the final response.

⇄ Shared-key Resolution (response, stage 1) `dof-2008-14-pdu-3`

Instance of Identity Resolution`dof-2008-16-pdu-3`



`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 the remaining fields.
`VALIDb`	**Forty bytes.** The value of VALID_B as discussed in the specification.

2.2. Storage

The authentication server must store information that allows identification using this mode. There is not a required format for this information, although there are guidelines on the information that should be stored.

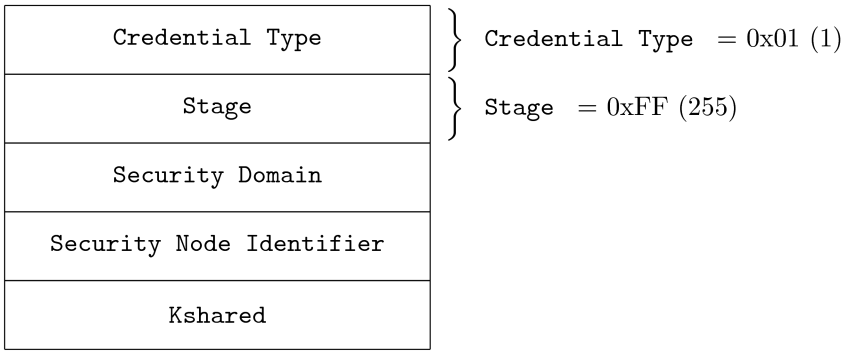
The storage format must include K_{SHARED}, the 256-bit shared key associated with the identity.

2.3. Persistence

It is often required to persist credential information. For example, a configuration file may need to contain a reference to a password. The reserved **Stage** value of 255 is used in this case as follows.

⇄ **Shared-key Persistence `dof-2008-14-pdu-4`**

Instance of Identity Persistence`dof-2008-16-pdu-4`



`Credential Type`	**Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.
`Stage`	**One byte.** The stage of the request.
`Security Domain`	**Instance of** Security Domain`dof-2008-16-pdu-5`.The security domain.
`Security Node Identifier`	**Instance of** Security Node Identifier`dof-2008-16-pdu-1`.Contains the identity as known by the authentication server.
`Kshared`	**Thirty-two bytes.** The shared key.

2.4. Shared-key Cryptography

The resolution of shared keys is not required except for domain discovery. In this case, both the node and the AS already share a key, and just need to prove their knowledge to each other.

RND	An 8-byte cryptographic random number.
A || B || C	Binary concatenation.
MAC[k](msg)	HMAC using SHA-256, where "k" is the security key and "msg" is the message to digest.
K_{SHARED}	The shared key between the AS and the node.
D	The domain of the responder.

The initial request includes a message and MAC.

msg_A = RND

$VALID_A = msg_A$ || MAC[K_{SHARED}](msg_A)

The AS validates this result and then passes out its own validator and returns it.

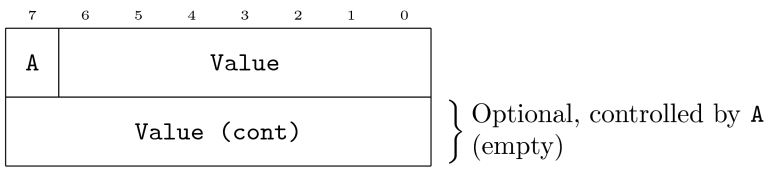
msg_B = RND

$VALID_B = msg_B$ || MAC[K_{SHARED}](msg_A || msg_B || D)

3. External PDU Reference

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

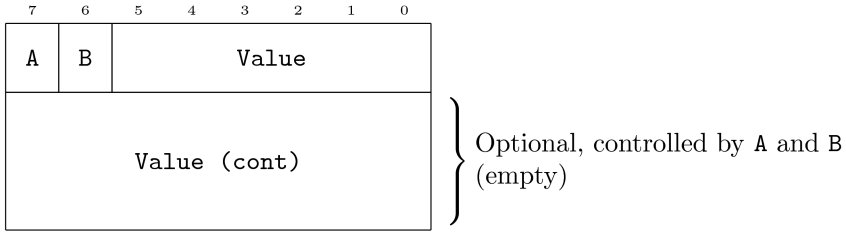
Compressed Unsigned 16-Bit `dof-2009-1-pdu-1`



`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`

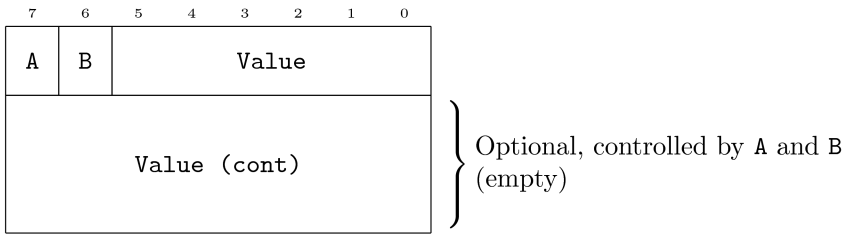


`A` **One bit.** Controls the size of `Value` and the presence of the `B` field.

`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.

Compressed Unsigned 32-Bit `dof-2009-1-pdu-3`

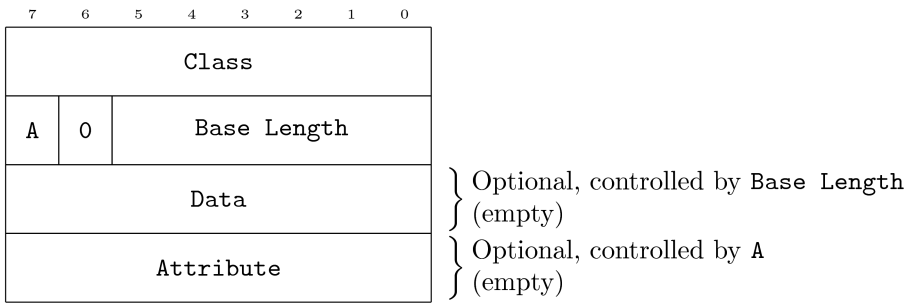


`A` **One bit.** Controls the size of `Value` and the presence of the `B` field.

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

`Value` **Either seven, fourteen, or thirty 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 thirty bits (four bytes) are used.

Object Identifier (OID) `dof-2009-1-pdu-4`



`Class` **Instance of** Compressed Unsigned 32-Bit`dof-2009-1-pdu-3`.The OID class, registered with ODP-TSC.

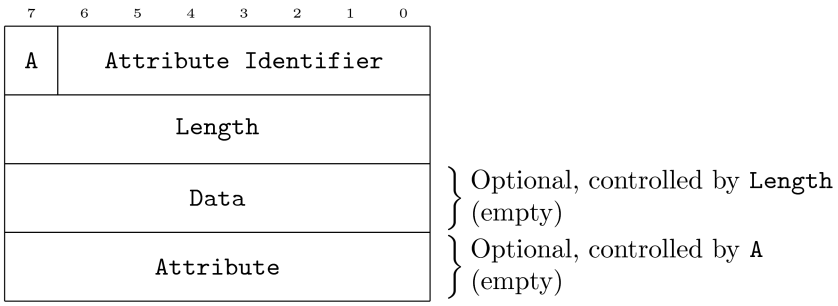
`A` **One bit.** Controls the presence of `Attribute`.

`Base Length` **Six bits.** This contains the length of the `Data`.

`Data` **Variable bytes.** The data associated with the OID. `Base Length` bytes.

`Attribute` **Instance of** Object Identifier Attribute`dof-2009-1-pdu-5`.**Optional, controlled by** `A`. If `A` is one then this field is present. Attributes form an ordered list, with each attribute containing a flag controlling the presence of the next attribute.

⇄ **Object Identifier Attribute** `dof-2009-1-pdu-5`



`A` **One bit.** Controls the presence of `Attribute`.

`Attribute Identifier` **Seven bits.** The attribute type, registered with ODP-TSC.

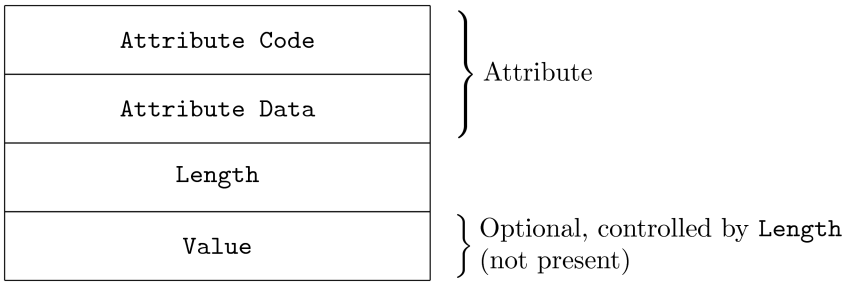
`Length` **One byte.** This field contains the length of the `Data`.

`Data` **Variable bytes.** The data associated with the attribute. `Length` bytes.

`Attribute` **Instance of** Object Identifier Attribute`dof-2009-1-pdu-5`.**Optional, controlled by** `A`. If `A` is one then this field is present. Attributes form an ordered list, with each attribute containing a flag controlling the presence of the next attribute.

3.2. DOF Protocol Specification, OpenDOF TSC, [dof-2008-1] (7.0.1, 12 January 2018)

⇄ **Attribute/Value Pair** `dof-2008-1-pdu-16`



`Attribute Code` **One byte.** This field identifies the attribute. It must be registered with the ODP-TSC.

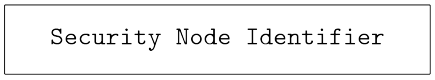
`Attribute Data` **Two bytes, MSB.** This field completes the definition of an attribute in combination with the code.

`Length`	**One byte.** This field determines the length of the `Value`. It may be zero, in which case the `Value` field is not present.
`Value`	**Variable length. Optional, controlled by** `Length`. Contains the value for the associated attribute.

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

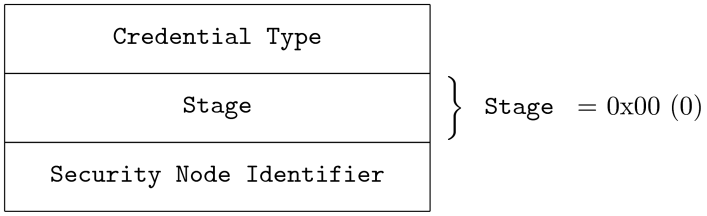
⇄ Security Node Identifier `dof-2008-16-pdu-1`

Instance of Object Identifier (OID)`dof-2009-1-pdu-4`



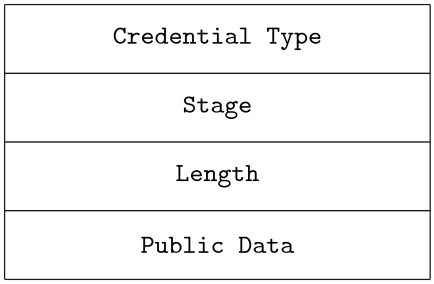
`Security Node Identifier`	**Instance of** Object Identifier (OID)`dof-2009-1-pdu-4`.This represents the node. It must not have attributes. It must not be broadcast or multicast.

⇄ Base Credential Format `dof-2008-16-pdu-2`



`Credential Type`	**Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.
`Stage`	**One byte.** The stage of the request. The indicated value is required. This value identifies the PDU as not including a length. The length of the **Security Node Identifier** is contained inside itself, any additional length would be redundant.
`Security Node Identifier`	**Instance of** Security Node Identifier`dof-2008-16-pdu-1`.Contains the identity as known by the authentication server.

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



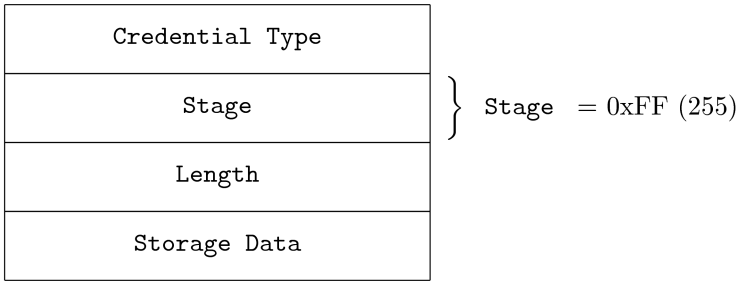
`Credential Type` **Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.

`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**.

Identity Persistence `dof-2008-16-pdu-4`



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

`Stage` **One byte.** The stage of the request.

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

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

Security Domain `dof-2008-16-pdu-5`

Instance of Object Identifier (OID)`dof-2009-1-pdu-4`



`Security Domain` **Instance of** Object Identifier (OID)`dof-2009-1-pdu-4`.This represents the domain. It must not have attributes. It must not be multicast. Certain PDUs allow the use of broadcast for the domain.