DOF OID Formats

DOF Protocol Specification

Technical Steering Committee

DOF OID Formats: DOF Protocol Specification

Technical Steering Committee

Document Information

Document Tracking Number	dof-2010-1
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 OID Formats, OpenDOF TSC, [dof-2010-1] (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. Introduction 9
3. OID Class Formats 11
    3.1. MAC-48 Address 11
        3.1.1. Data Format 11
        3.1.2. Usage Requirements 12
    3.2. E-Mail 12
        3.2.1. Data Format 12
        3.2.2. Usage Requirements 13
    3.3. IPv4 Address 14
        3.3.1. Data Format 14
        3.3.2. Usage Requirements 14
    3.4. IPv6 Address 15
        3.4.1. Data Format 15
        3.4.2. Usage Requirements 16
    3.5. Fully-qualified Domain Names 16
        3.5.1. Data Format 16
        3.5.2. Usage Requirements 17
    3.6. EUI-48™ Identifier 17
        3.6.1. Data Format 17
        3.6.2. Usage Requirements 18
    3.7. EUI-64™ Identifier 18
        3.7.1. Data Format 18
        3.7.2. Usage Requirements 19
    3.8. Smart Energy System Serial Number (SANYO, Panasonic Corporation) 19
        3.8.1. Data Format 20
        3.8.2. Usage Requirements 21
    3.9. AWS Resource ID (Amazon Web Services) 21
        3.9.1. Data Format 21
        3.9.2. Usage Requirements 22
    3.10. ODP-TSC Assigned String Format 22
        3.10.1. Data Format 22
        3.10.2. Usage Requirements 23
4. OID Attribute Formats 25
    4.1. Provider 25
    4.2. Group 25
    4.3. Binding Provider 26
    4.4. Related Date/Time 27
    4.5. Created Date/Time 28

5. External PDU Reference 29
5.1. DOF Common Types, OpenDOF TSC, [dof-2009-1] (7.0.1, 12 January 2018) 29

Specifications and PDUs

MAC-48 Address `dof-2010-1-pdu-1` 11
Email Address `dof-2010-1-pdu-2` 12
IPv4 Address `dof-2010-1-pdu-3` 14
IPv6 Address `dof-2010-1-pdu-4` 15
Domain `dof-2010-1-pdu-5` 16
EUI-48 Address `dof-2010-1-pdu-6` 17
EUI-64 Address `dof-2010-1-pdu-7` 18
Smart Energy System Serial Number `dof-2010-1-pdu-8` 20
AWS Resource ID `dof-2010-1-pdu-9` 21
ODP-TSC Assigned String Format `dof-2010-1-pdu-10` 22
Provider Attribute `dof-2010-1-pdu-11` 25
Group Attribute `dof-2010-1-pdu-12` 26
Binding Provider Attribute `dof-2010-1-pdu-13` 26
Related Date/Time Attribute `dof-2010-1-pdu-14` 27
Created Date/Time Attribute `dof-2010-1-pdu-15` 28
Compressed Unsigned 16-Bit `dof-2009-1-pdu-1` 29
Compressed Unsigned 32-Bit `dof-2009-1-pdu-3` 29
Object Identifier (OID) `dof-2009-1-pdu-4` 30
Object Identifier Attribute `dof-2009-1-pdu-5` 30
Interface Identifier (IID) `dof-2009-1-pdu-21` 31

1. About This Document

This document contains reference information for DOF OID formats. The OID is a primary type in the DOF object model. It is an extensible type, with its base defined in the *DOF Common Types* specification. The document also contains formats for some registered OID Attribute Identifiers.

Each OID format is defined by an OID Class, which is registered with the Technical Steering Committee of the OpenDOF Project, Inc. (ODP-TSC) before use. This document contains many of these defined formats, but not all of them. The registry of OID Class assignments (found at https://opendof.org/registry-oid-class-odp-tsc) and the registry of OID Attribute Identifier assignments (found at https://opendof.org/registry-oid-attribute) contain links to other defining documents.

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.

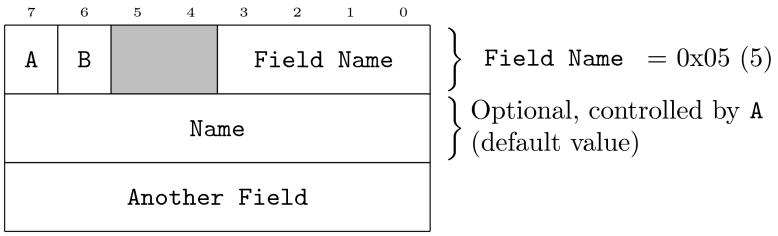
1.3.4. Fields

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. Introduction

Each OID Class is required to define its format. The format extends a standard PDU defined in *DOF Common Types*.

This document contains many of these additional class definitions. Each section presents a class, the structure of its data, and the requirements that the data must meet in order to ensure uniqueness.

All use of these formats must adhere to the requirements specified for the class. Failure to meet the requirements is a serious error that can affect interoperability and uniqueness of DOF systems.



The types described in this document are independent of protocol versioning. They are common among all DOF protocol versions, and are fixed. They can be added to over time, and protocols that use those new types can be negotiated if necessary according to the DOF Protocol Stack specification.

Throughout this document are references to "ODP-TSC" which is the Technical Steering Committee of the OpenDOF Project, Inc. Contact information for the ODP-TSC is at the beginning of this document (inside cover page).

3. OID Class Formats

The following sections define some of the registered OID Class formats. For the formats defined here this is the authoritative reference, all implementations must meet the specified formats and use rules.

3.1. MAC-48 Address

This OID class uses a MAC-48 address to provide uniqueness.



The MAC-48 class is similar to the EUI-48™ format. The encapsulated data follows the same format for the most part. Note however that the two OID classes form different, non-equivalent OID spaces. The appropriate OID class should be used to ensure comparisons are correct. The correct class depends on the definition of the underlying data used. The underlying data specification should indicate whether the data is MAC-48 or EUI-48™. If the underlying data is unclear, then the MAC-48 class should be used.

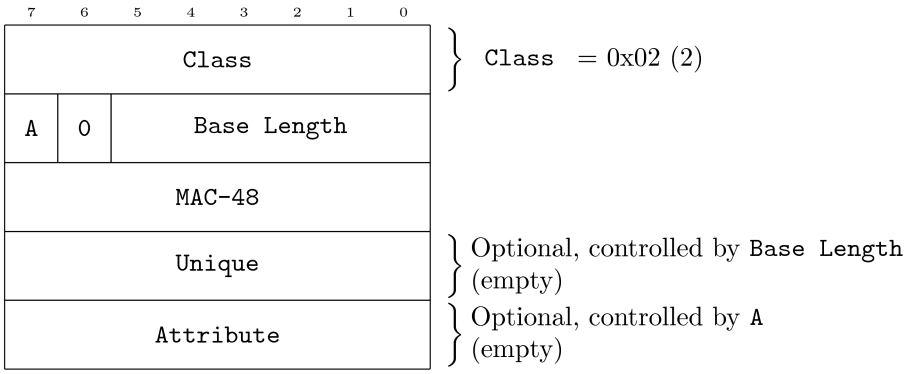
3.1.1. Data Format

The data format contains the unique MAC-48 address and optional additional data. The "owner" of the MAC-48 address can use the additional data to create multiple OIDs from the same MAC-48 address.



MAC-48 Address `dof-2010-1-pdu-1`

Instance of 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 `MAC-48` and `Unique` fields and must be at least six.
`MAC-48`	**Six bytes.** The MAC-48, MSB first.

`Unique`	**Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same MAC. This is an object index, MSB first. Leading zero bytes are not allowed.
`Attribute`	**Instance of** Object Identifier Attribute$_{\texttt{dof-2009-1-pdu-5}}$.**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.

3.1.2. Usage Requirements

This format must only be used with unicast, globally unique (OUI enforced) values. This means that the low two bits of the OUI high byte will be zero (true of all registered OUI values).

Only the "owner" of the MAC-48 address may use this format. The hardware that the MAC-48 address was assigned to is the "owner," and by extension the software running on that hardware. This class is most useful to embedded devices that base their OID on their physical hardware.

MAC-48 address assignments are permanent, and so this OID class is temporally stable.

The MAC-48 address used must use an OUI registered with IEEE.

3.2. E-Mail

This class uses an e-mail address to provide uniqueness.

This class can be useful for identifying users for login, where the application requests the user enter a username. However, as mentioned above this possible use does not limit the other potential uses of this class of OID. Specifically, this is not the only class of OID that can identify a user and its use does not imply user identification.

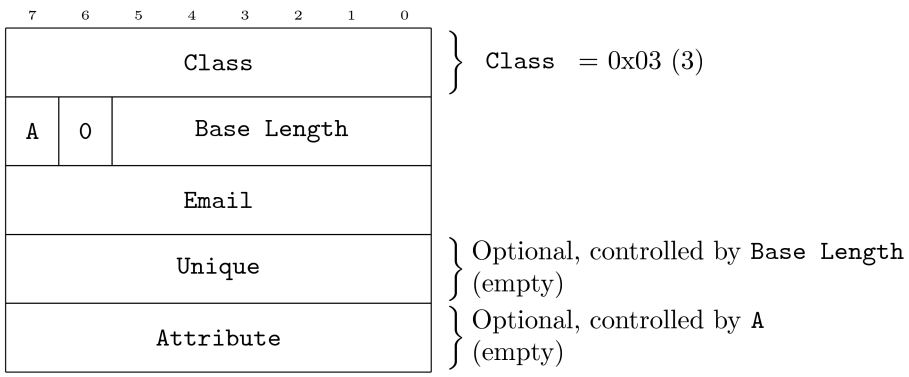
3.2.1. Data Format

This data contains the e-mail address used. Any upper-case characters should be converted to lower-case. Other requirements are specified in RFC2822, section 3.4.1, "addr-spec". The string must not contain a null terminator.

To use the same e-mail address to create multiple OIDs, a unique suffix can be added, but it must begin with an ASCII colon and contain only ASCII characters.

⇄ **Email Address** **`dof-2010-1-pdu-2`**

Instance of Object Identifier (OID)$_{\texttt{dof-2009-1-pdu-4}}$



`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 `Email` and `Unique` fields and must be at least five.

`Email` **Variable characters.** The email address as specified in RFC2822, section 3.4.1, "addr-spec".

`Unique` **Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same email. This is a decimal object index in US-ASCII (only "0" through "9" and ":" are allowed. Leading zeros are not allowed. If present, this field begins with the character ":" (colon).

`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.

3.2.2. Usage Requirements

Only public email addresses should be used. This requires that the domain used in the e-mail must be registered to the organization referred to.

It would be extremely rare to use this OID class with additional data, although it is supported. The reason is that uniqueness could be very difficult to ensure in this case – the program would have a hard time knowing that the same e-mail was not being used elsewhere.

Because of this, this OID class is most used during registration processes and then later during lookups rather than in situations where new OIDs are being generated.

E-mail addresses are also not assigned permanently. This means that they should not be used in situations where they cannot be invalidated if ownership of the address changes. The same applies to the domain name used in the e-mail address.

3.3. IPv4 Address

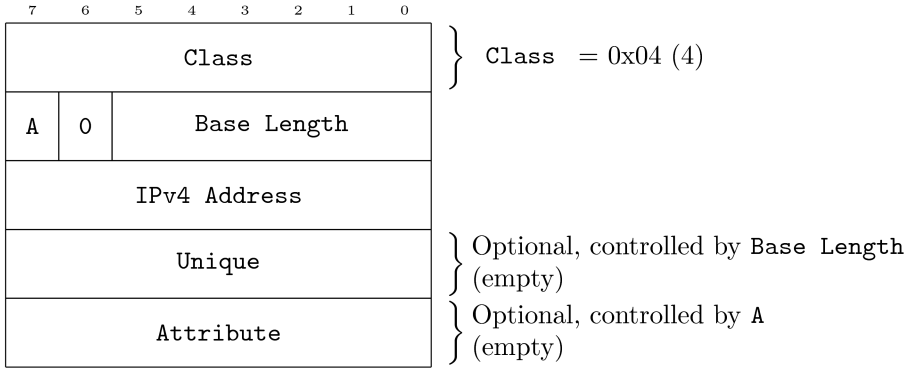
This class uses an IPv4 address to provide uniqueness.

3.3.1. Data Format

The data contains the IPv4 address and optional additional data.

⇄ **IPv4 Address** `dof-2010-1-pdu-3`

Instance of 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 `IPv4 Address` and `Unique` fields and must be at least four.

`IPv4 Address` **Four bytes.** The IPv4 address, MSB first.

`Unique` **Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same IPv4 address. This is an object index, MSB first. Leading zero bytes are not allowed.

`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.

3.3.2. Usage Requirements

This class cannot be used for local (for example, 10.x.x.x) addresses because this would allow possible aliasing between organizations (OIDs are *globally* unique). The address must be stable and unique.

Note that IPv4 addresses are not permanently assigned, and so the use of this class is restricted to temporary uses.

3.4. IPv6 Address

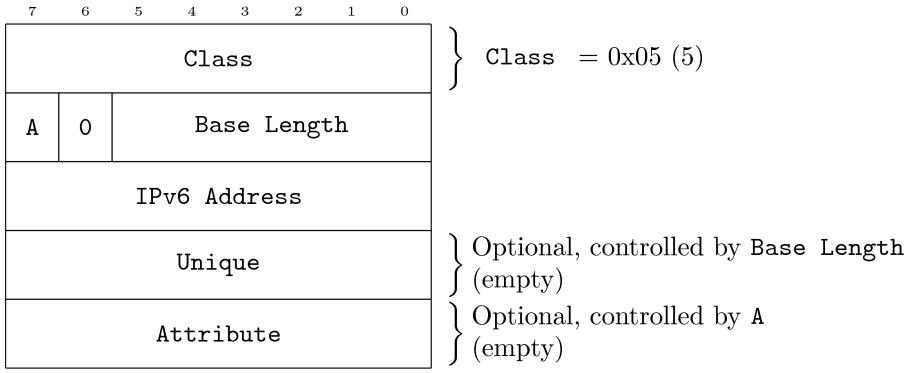
This class uses an IPv6 address to provide uniqueness.

3.4.1. Data Format

The data contains the IPv6 address and optional additional data. . The data is passed MSB first.

IPv6 Address `dof-2010-1-pdu-4`

Instance of 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 `IPv6 Address` and `Unique` fields and must be at least sixteen.

`IPv6 Address` **Sixteen bytes.** The IPv4 address, MSB first.

`Unique` **Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same IPv6 address. This is an object index, MSB first. Leading zero bytes are not allowed.

`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.

3.4.2. Usage Requirements

This class cannot be used with site- or link-local addresses because this would allow possible aliasing between organizations (OIDs are *globally* unique). The address must be stable and unique.

Note that IPv6 addresses are not permanently assigned, and so the use of this class is restricted to temporary uses.

3.5. Fully-qualified Domain Names

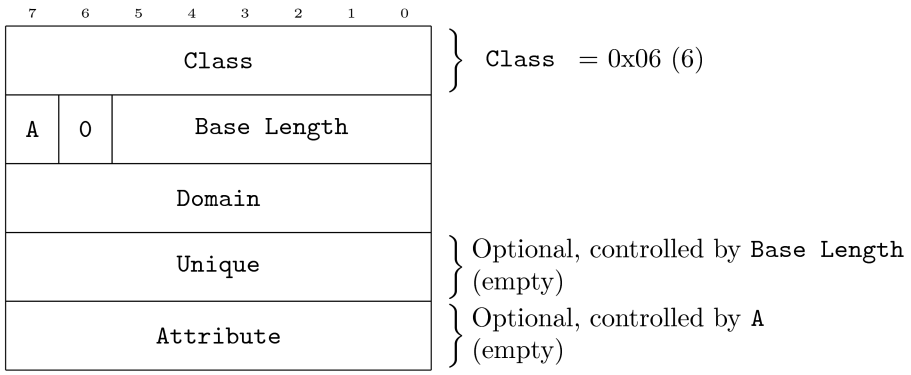
This class format uses a fully-qualified domain name to provide uniqueness.

3.5.1. Data Format

The data contains the fully-qualified domain name (FQDN) in ASCII and optional additional data. Upper-case characters must be converted to lower-case. There is no null terminator. If necessary a unique suffix can be added, but it must begin with an ASCII colon and contain only ASCII characters.

Domain `dof-2010-1-pdu-5`

Instance of 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 `Domain` and `Unique` fields and must be at least three.

`Domain` **Variable characters.** The domain as specified in RFC2822, "dot-atom".

`Unique` **Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same domain. This is a decimal object index in US-ASCII (only "0" through "9" and ":" are allowed. Leading zeros are not allowed. If present, this field begins with the character ":" (colon).

`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.

3.5.2. Usage Requirements

FQDN assignments are not permanent, and so this OID class is limited to cases where it can be changed if the assignment is no longer valid. This class can only be used by the registered owner of the domain name. The registered owner can also use any subdomains that use the registered base name.

3.6. EUI-48™ Identifier

This class encapsulates EUI-48™ identifiers.

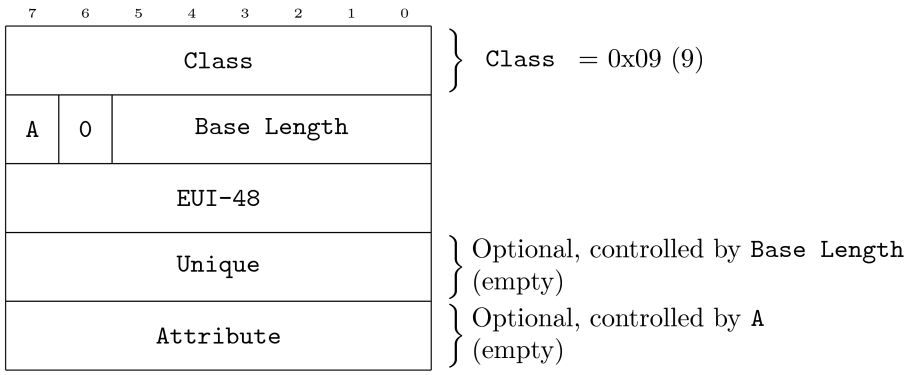
The MAC-48 class is similar to the EUI-48™ format. The encapsulated data follows the same format for the most part. Note however that the two OID classes form different, non-equivalent OID spaces. The appropriate OID class should be used to ensure comparisons are correct. The correct class depends on the definition of the underlying data used. The underlying data specification should indicate whether the data is MAC-48 or EUI-48™. If the underlying data is unclear, then the MAC-48 class should be used.

3.6.1. Data Format

The data contains EUI-48™ identifier and optional additional data. The standard (numeric) format is used.

EUI-48 Address `dof-2010-1-pdu-6`

Instance of 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 `EUI-48` and `Unique` fields and must be at least six.

`EUI-48`	**Six bytes.** The EUI-48, MSB first.
`Unique`	**Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same EUI-48. This is an object index, MSB first. Leading zero bytes are not allowed.
`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.

3.6.2. Usage Requirements

This format must only be used with unicast, globally unique (OUI enforced) values. This means that the low two bits of the OUI high byte will be zero (true of all registered OUI values).

Only the "owner" of the EUI-48™ address may use this format. The hardware that the EUI-48™ address was assigned to is the "owner," and by extension the software running on that hardware. This class is most useful to embedded devices that base their OID on their physical hardware.

Only permanently assigned EUI-48™ identifiers may be used, and so this OID class is temporally stable.

The EUI-48™ address used must use an OUI registered with IEEE.

3.7. EUI-64™ Identifier

This class encapsulates EUI-64™ identifiers.



The EUI-64™ identifier can encapsulate both MAC-48 and EUI-48™ identifiers. However, OID class formats for all of these classes form individual OID spaces that are not equivalent. Applications must decide whether or not to use the larger EUI-64™ identifier format and must do so consistently when encapsulation is used.

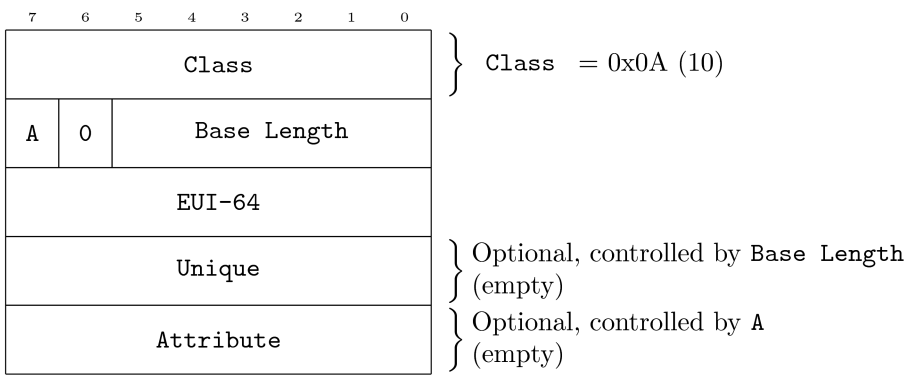
3.7.1. Data Format

The data contains EUI-64™ identifier and optional additional data. The standard (numeric) format is used.



EUI-64 Address `dof-2010-1-pdu-7`

Instance of 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 `EUI-64` and `Unique` fields and must be at least eight.

`EUI-64` **Eight bytes.** The EUI-64, MSB first.

`Unique` **Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same EUI-64. This is an object index, MSB first. Leading zero bytes are not allowed.

`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.

3.7.2. Usage Requirements

This format must only be used with unicast, globally unique (OUI enforced) values. This means that the low two bits of the OUI high byte will be zero (true of all registered OUI values).

Only the "owner" of the EUI-64™ address may use this format. The hardware that the EUI-64™ address was assigned to is the "owner," and by extension the software running on that hardware. This class is most useful to embedded devices that base their OID on their physical hardware.

Only permanently assigned EUI-64™ identifiers may be used, and so this OID class is temporally stable.

The EUI-64™ address used must use an OUI registered with IEEE.

3.8. Smart Energy System Serial Number (SANYO, Panasonic Corporation)

This class is used for devices produced by Sanyo Corporation. Sanyo has stated that their serial numbers are globally unique. This is done through a combination of product code and manufacture date. The format

may be understood more thoroughly by Sanyo, although the detailed specification of any information contained within the serial number (like a product code) is not defined here.

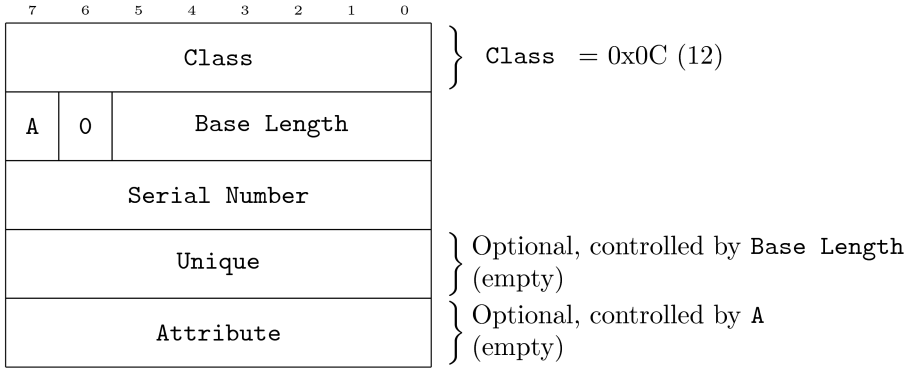
3.8.1. Data Format

The data contains a case-sensitive ASCII string. The string must not contain a null terminator. The serial number (not including any additional 'unique' information) contains a maximum of 32 characters. Non-printing characters are not allowed.

It is possible to create multiple object identifiers from the same Sanyo serial number. This is done by using the serial number as a base, and adding unique content to the end. To use the same serial number to create multiple OIDs, a unique ASCII numeric suffix can be added, beginning with an ASCII TAB and containing only ASCII characters "0" through "9" with no leading zeroes. This means that the ASCII TAB character separates the Sanyo serial number and the unique suffix. The ASCII TAB character is not allowed in the Sanyo serial number.

Smart Energy System Serial Number `dof-2010-1-pdu-8`

Instance of 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 `Serial Number` and `Unique` fields and must be at least five.

`Serial Number` **Variable characters.** The Sanyo serial number. This field has a maximum length of 32 bytes, no null terminator, and must not include non-printing characters.

`Unique` **Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same serial number. This is a decimal index in US-ASCII (only "0" through "9") with an initial TAB are allowed. Leading zeros are not allowed. If present, this field must begin with the character TAB (ASCII 7).

`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.

3.8.2. Usage Requirements

The smart energy business unit ensures the uniqueness of their serial numbers.

3.9. AWS Resource ID (Amazon Web Services)

This class can be used for any resource defined by AWS, including instances. The instance ID (instance resource ID) of a running instance is easily obtained, and useful as a unique identifier.

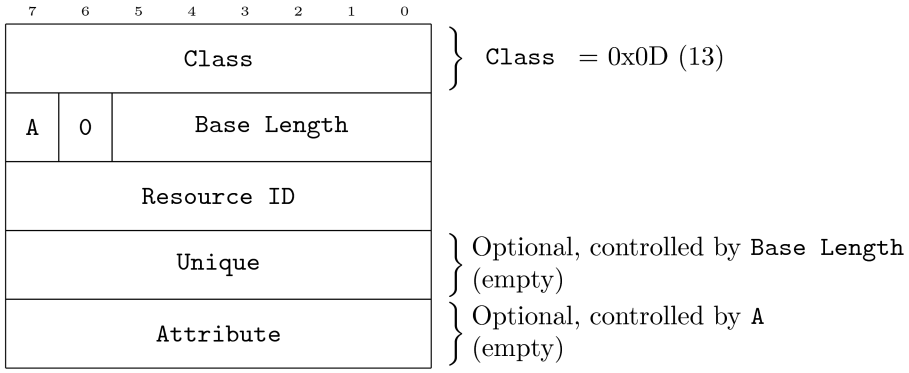
3.9.1. Data Format

The data contains a case-sensitive ASCII string. The string must not contain a null terminator. The serial number (not including any additional 'unique' information) contains a maximum of 32 characters. Non-printing characters are not allowed. The formatting is exactly as returned by the AWS APIs.

It is possible to create multiple object identifiers from the same AWS resource ID. This is done by using the ID as a base, and adding unique content to the end. To use the same serial number to create multiple OIDs, a unique ASCII numeric suffix can be added, beginning with an ASCII TAB and containing only ASCII characters "0" through "9" with no leading zeroes. This means that the ASCII TAB character separates the AWS resource ID and the unique suffix. The ASCII TAB character is not allowed in the AWS resource ID.

⇄ **AWS Resource ID `dof-2010-1-pdu-9`**

Instance of 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 `Resource ID` and `Unique` fields.

`Resource ID` **Variable characters.** The AWS Resource ID. This field has a maximum length of 32 bytes.

`Unique` **Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same resource ID. This is a decimal index in US-ASCII (only "0" through "9") with an initial TAB are allowed. Leading zeros are not allowed. If present, this field must begin with the character TAB (ASCII 7).

`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.

3.9.2. Usage Requirements

Amazon has stated in numerous forums that AWS Resource IDs are globally unique. It is not clear whether they are ever reused, but if they are the assumption is that they would refer to the same "object".

3.10. ODP-TSC Assigned String Format

This class uses an ODP-TSC assigned identifier to provide a base for uniqueness, and allows the owner of the identifier to create as many string-based identifiers as desired. This is similar to class 127, but uses a string instead of a binary format.

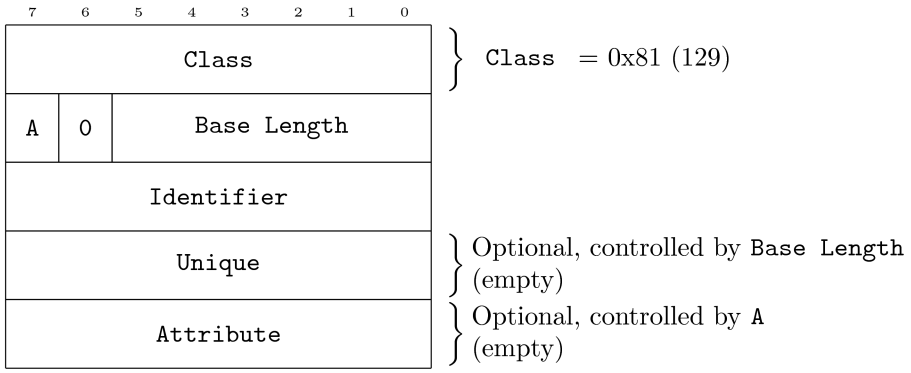
3.10.1. Data Format

This data contains the ODP-TSC assigned identifier in addition to any additional unique information. The string must not contain a null terminator.

To use the same identifier to create multiple OIDs, a unique suffix can be added, but it must begin with an ASCII '/' and contain only ASCII characters.

⇄ **ODP-TSC Assigned String Format** `dof-2010-1-pdu-10`

Instance of 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 `Identifier` and `Unique` fields and must be at least five.
`Identifier`	**Variable characters.** The ODP-TSC assigned identifier, represented as upper-case hex digits with no leading zeros (even if this results in an odd number of digits).
`Unique`	**Variable length. Optional, controlled by** `Base Length`. Optional data associated with the OID to uniquely identify objects that share the same base identifier. The data is ASCII. If present, this field begins with the character "/" (forward slash).
`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.

3.10.2. Usage Requirements

The owner of the ODP-TSC assigned identifier is responsible to ensure uniqueness. This format is most commonly used when a single assigned identifier is used as a base, with unique data being added to create a set of related identifiers. Up to the length of the OID data as much required unique data as is needed can be added.

It is recommended, but not required, that different groupings of unique data use the slash character ('/') as a separator.

4. OID Attribute Formats

The following sections define some of the registered OID Attribute Identifier formats. For the formats defined here this is the authoritative reference, all implementations must meet the specified formats and use rules.

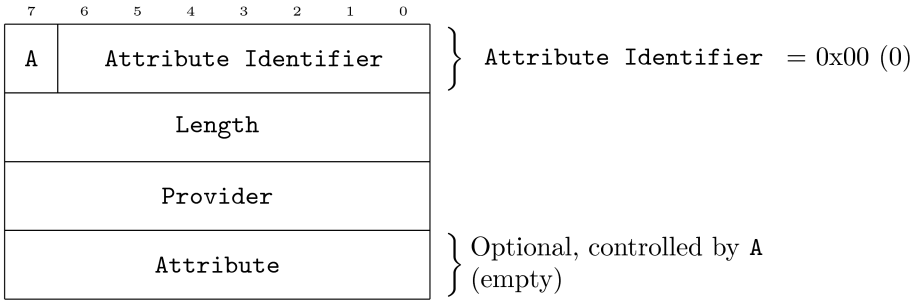
4.1. Provider

This Attribute identifies the provider of a service. It can be used in cases where a base OID is already known, but the provider of an object desires to uniquely identify itself in addition. This is a common use case for services that extend an existing object, but want to co-exist with other potential services providing the same bindings. Without using this Attribute, such services would "collide" with one another.



Provider Attribute `dof-2010-1-pdu-11`

Instance of 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 `Provider`.
`Provider`	**Instance of** Object Identifier (OID)`dof-2009-1-pdu-4`.The Object Identifier of the provider.
`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.

4.2. Group

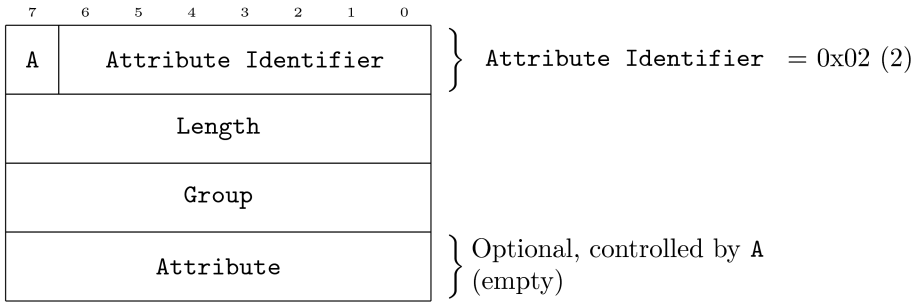
This Attribute is used to create a general group. Members identify themselves by including this attribute with data that identifies the group.

This can be used in an object model to group objects, but any Attribute can serve the same purpose.

The "meaning" of using this Attribute depends on the meaning of the identified group, however an Attribute can only be used a single time in an object. This requirement places limits on the general uses of this Attribute.

Group Attribute `dof-2010-1-pdu-12`

Instance of Object Identifier Attribute`dof-2009-1-pdu-5`



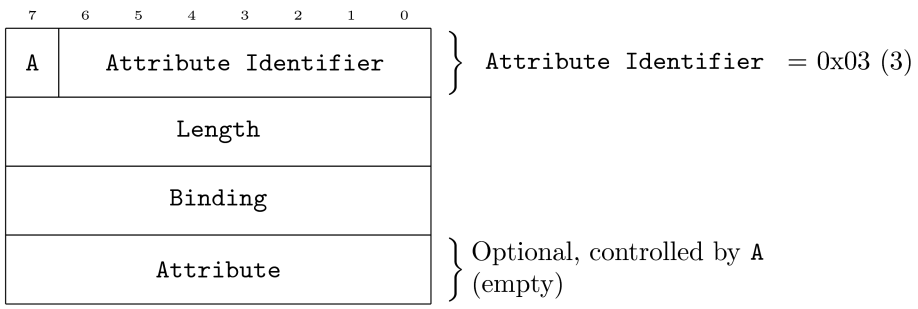
`A`	**One bit.** Controls the presence of `Attribute`.
`Attribute Identifier`	**Seven bits.** The attribute type, registered with ODP-TSC.
`Length`	**One byte.** This field contains the length of the `Group`.
`Group`	**Instance of** Object Identifier (OID)`dof-2009-1-pdu-4`.The Object Identifier of the group.
`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.

4.3. Binding Provider

This Attribute identifies the provider of a binding. It can be used in cases where the provider of a specific binding wants to provide additional related bindings. Note that in the DOF architecture each interface may have a unique provider, and so the binding provider applies to a single interface.

Binding Provider Attribute `dof-2010-1-pdu-13`

Instance of Object Identifier Attribute`dof-2009-1-pdu-5`



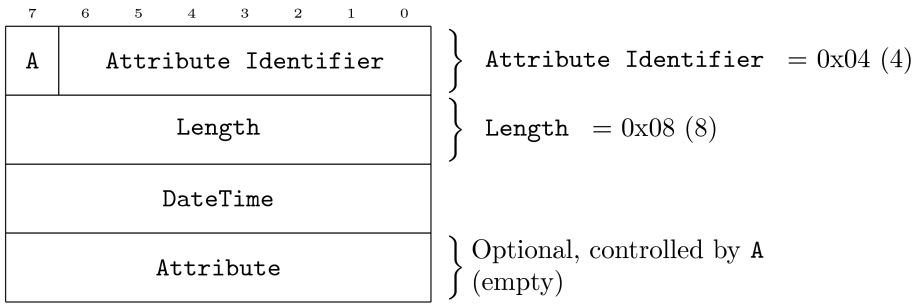
`A`	**One bit.** Controls the presence of `Attribute`.
`Attribute Identifier`	**Seven bits.** The attribute type, registered with ODP-TSC.
`Length`	**One byte.** This field contains the length of the `Binding`.
`Binding`	**Instance of** Interface Identifier (IID)`dof-2009-1-pdu-21`.The Interface Identifier of the binding.
`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.

4.4. Related Date/Time

For objects that are associated with a specific date and time, this attribute can express that relationship directly as part of the OID.

⇄ Related Date/Time Attribute `dof-2010-1-pdu-14`

Instance of 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.

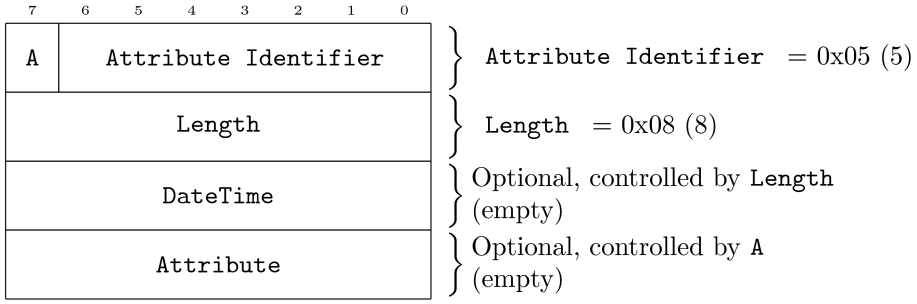
`Length`	**One byte.** This field contains the length of the `DateTime`.
`DateTime`	**Eight bytes.** Follows the wire definition of Type 0x40, Date/Time.
`Attribute`	**Instance of** Object Identifier Attribute`dof-2009-1-pdu-5`.**Optional, controlled by** `A`. If `A` is one then this field is present. Attributes form an ordered list, with each attribute containing a flag controlling the presence of the next attribute.

4.5. Created Date/Time

For objects that are created at a specific date and time, this attribute can express that relationship directly as part of the OID.

⇄ Created Date/Time Attribute `dof-2010-1-pdu-15`

Instance of Object Identifier Attribute`dof-2009-1-pdu-5`

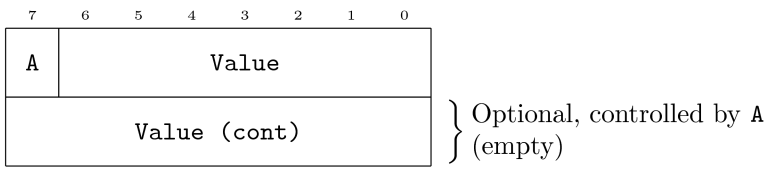


`A`	**One bit.** Controls the presence of `Attribute`.
`Attribute Identifier`	**Seven bits.** The attribute type, registered with ODP-TSC.
`Length`	**One byte.** This field contains the length of the `DateTime`.
`DateTime`	**Eight bytes.** Follows the wire definition of Type 0x40, Date/Time.
`Attribute`	**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.

5. External PDU Reference

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

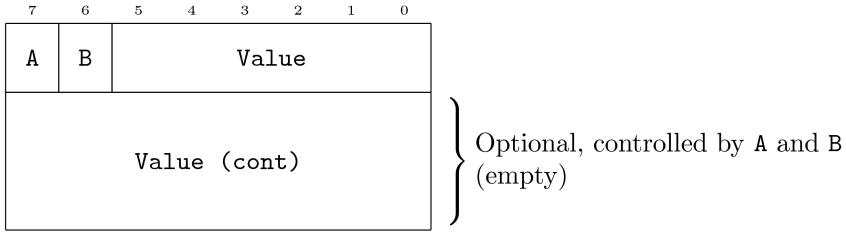
Compressed Unsigned 16-Bit `dof-2009-1-pdu-1`



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

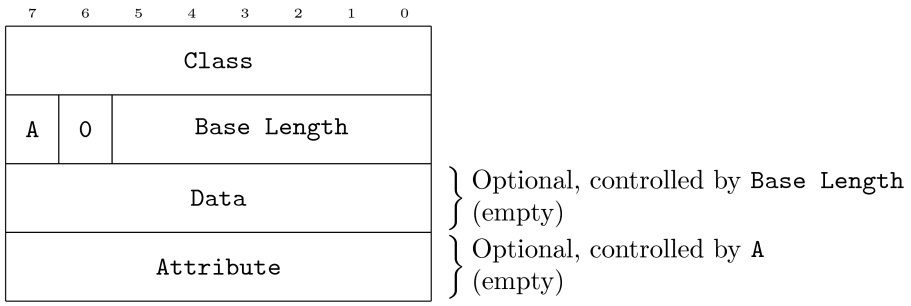


`A` **One bit.** Controls the size of `Value` and the presence of the `B` field.

`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.

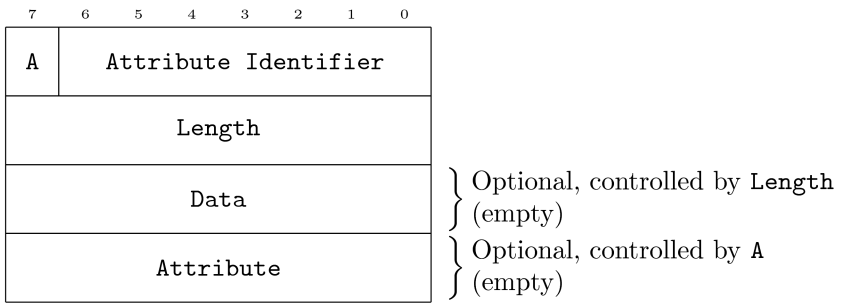
`A` **One bit.** Controls the presence of `Attribute`.

`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`.

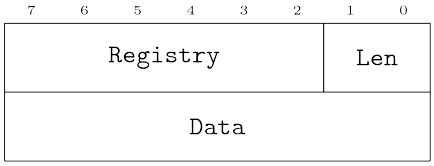
`Attribute Identifier` **Seven bits.** The attribute type, registered with ODP-TSC.

`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.

⇄ Interface Identifier (IID) `dof-2009-1-pdu-21`



`Registry` **Six bits.** The registry of the IID, registered with the ODP-TSC.

`Len` **Two bits.** Defines the length of the `Data`, which is $2^{(LEN-1)}$. `Len` equal to zero (0) is invalid.

`Data` **Variable length, controlled by `Len`.** The identifier for the IID within the `Registry`.