Ticket Request Protocol (V1)

DOF Protocol Specification

Technical Steering Committee

Ticket Request Protocol (V1): DOF Protocol Specification

Technical Steering Committee

Document Information

Document Tracking Number	dof-2008-5
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	Ticket Request Protocol (V1), OpenDOF TSC, [dof-2008-5] (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. Overview 9
    2.1. Operation State vs. Authentication Server State 10
    2.2. Local and Remote Domains 10
3. Specification 11
    3.1. Application Format 11
    3.2. Security Primitives 12
    3.3. Negotiation 12
    3.4. Multicast Behavior 13
    3.5. Authentication Server Discovery 13
    3.6. Domain Discovery 13
    3.7. Routing 14
    3.8. Session Requirements 15
    3.9. Operation State Requirements 15
    3.10. Security Considerations 15
    3.11. Request Types 16
4. Command Reference 17
    4.1. Reject 17
        4.1.1. Flows 18
        4.1.2. Details 18
    4.2. Request KEK 20
        4.2.1. Flows 23
        4.2.2. Details 23
        4.2.3. Cryptography 24
    4.3. Request Local Domain 24
        4.3.1. Flows 26
        4.3.2. Details 26
        4.3.3. Cryptography 27
    4.4. Request Random 28
        4.4.1. Flows 29
        4.4.2. Details 29
        4.4.3. Cryptography 29
    4.5. Request Remote Domain 30
        4.5.1. Flows 32
        4.5.2. Details 32
        4.5.3. Cryptography 33
    4.6. Request Security Scopes 34
        4.6.1. Flows 36
        4.6.2. Details 36

4.6.3. Cryptography 37
4.7. Request Session 37
4.7.1. Flows 39
4.7.2. Details 39
4.7.3. Cryptography 40
4.8. Resolve Credential 40
4.8.1. Flows 42
4.8.2. Details 42
4.8.3. Routing 43
4.9. Validate Credential 43
4.9.1. Flows 44
4.9.2. Details 44
4.9.3. Shared-key Cryptography 45
5. Sequence Diagrams 47
5.1. Stateless Command/Response 47
5.2. Stateful Multiple Command/Response 47
5.3. Local/Remote Domain 48
6. External PDU Reference 49
6.1. DOF Common Types, OpenDOF TSC, [dof-2009-1] (7.0.1, 12 January 2018) 49
6.2. DOF Protocol Specification, OpenDOF TSC, [dof-2008-1] (7.0.1, 12 January 2018) 52
6.3. DOF Security Specification, OpenDOF TSC, [dof-2008-16] (7.0.1, 12 January 2018) 54

List of Figures

5.1. Stateless Command/Response 47
5.2. Stateful Command/Response 47
5.3. Stateful Command/Response 48

List of Tables

4.1. TRP Error Codes 18
4.2. Request KEK Cryptography Parameters 24
4.3. Request Local Domain Cryptography Parameters 27
4.4. Request Random Cryptography Parameters 30
4.5. Request Remote Domain Cryptography Parameters 34
4.6. Request Security Scopes Cryptography Parameters 37
4.7. Request Session Cryptography Parameters 40

Specifications and PDUs

TRP Application Command Format `dof-2008-5-pdu-1` 11
TRP Application Response Format `dof-2008-5-pdu-2` 11
Nodes must allow TRP during authentication if it was negotiated in DSP. `dof-2008-5-spec-1` 12
DSP Authentication Attribute `dof-2008-5-pdu-3` 12
Respond correctly to multicast commands. `dof-2008-5-spec-2` 13
Keep routing paths the same for each command as much as possible. `dof-2008-5-spec-3` 14
Only forward to nodes that have been configured or validated. `dof-2008-5-spec-4` 15
Ensure that the PDUs used in TRP are forwarded without modification. `dof-2008-5-spec-5` 15
Reject `dof-2008-5-pdu-4` 17
Unregistered TRP Error Codes are not allowed. `dof-2008-5-spec-6` 20
Request KEK (command) `dof-2008-5-pdu-5` 20
Request KEK Request Block `dof-2008-5-pdu-6` 21
Request KEK (response) `dof-2008-5-pdu-7` 21
Request KEK Initialization Block `dof-2008-5-pdu-8` 22
Request Local Domain (command) `dof-2008-5-pdu-9` 24
Request Local Domain Request Block `dof-2008-5-pdu-10` 25
Request Local Domain (response) `dof-2008-5-pdu-11` 26
Request Local Domain Response Block `dof-2008-5-pdu-12` 26
Request Random (command) `dof-2008-5-pdu-13` 28
Request Random Request Block `dof-2008-5-pdu-14` 28
Request Random (response) `dof-2008-5-pdu-15` 28
Request Remote Domain (command) `dof-2008-5-pdu-16` 30
Request Remote Domain Request Block `dof-2008-5-pdu-17` 31
Request Remote Domain (response) `dof-2008-5-pdu-18` 31
Request Discovered Remote Domain (response) `dof-2008-5-pdu-19` 31
Request Remote Domain Response Block `dof-2008-5-pdu-20` 32
Request Security Scopes (command) `dof-2008-5-pdu-21` 34
Request Security Scopes Request Block `dof-2008-5-pdu-22` 35
Request Security Scopes (response) `dof-2008-5-pdu-23` 35
Request Security Scopes Initialization Block `dof-2008-5-pdu-24` 36
Request Session (command) `dof-2008-5-pdu-25` 37
Request Session (response) `dof-2008-5-pdu-26` 38
Resolve Credential (command) `dof-2008-5-pdu-27` 40
Resolve Credential (response) `dof-2008-5-pdu-28` 41
Resolve Discovered Credential (response) `dof-2008-5-pdu-29` 41
Validate Credential (command) `dof-2008-5-pdu-30` 43
Validate Credential (response) `dof-2008-5-pdu-31` 44
Compressed Unsigned 16-Bit `dof-2009-1-pdu-1` 49
Compressed Unsigned 24-Bit `dof-2009-1-pdu-2` 49
Compressed Unsigned 32-Bit `dof-2009-1-pdu-3` 50
Object Identifier (OID) `dof-2009-1-pdu-4` 50
Object Identifier Attribute `dof-2009-1-pdu-5` 51
AS Assigned `dof-2009-1-pdu-38` 51
General Application Header `dof-2008-1-pdu-13` 52
DSP Command/Response Format `dof-2008-1-pdu-15` 53
Attribute/Value Pair `dof-2008-1-pdu-16` 53
Configuration Negative Acknowledge `dof-2008-1-pdu-19` 53
Configuration Reject `dof-2008-1-pdu-20` 54

Attribute Data List `dof-2008-1-pdu-21` 54
Security Node Identifier `dof-2008-16-pdu-1` 54
Base Credential Format `dof-2008-16-pdu-2` 55
Identity Resolution `dof-2008-16-pdu-3` 55
Security Domain `dof-2008-16-pdu-5` 56
Security Scope `dof-2008-16-pdu-6` 56
Permission `dof-2008-16-pdu-7` 56
Permission Request `dof-2008-16-pdu-8` 57
Permission Validation `dof-2008-16-pdu-9` 57
Security Mode of Operation Negotiation `dof-2008-16-pdu-11` 57
Key Request `dof-2008-16-pdu-13` 58
Key Grant `dof-2008-16-pdu-14` 59
Session Initiator Block `dof-2008-16-pdu-15` 59
Session Responder Block `dof-2008-16-pdu-16` 59
Authentication Response Block `dof-2008-16-pdu-17` 60

1. About This Document

This document contains information about the Ticket Request Protocol, used with the DOF Protocol Stack (DPS). This is critical information for anyone who needs to work with the protocols themselves, and contains useful background information for those interested in the architecture and inner workings of DOF implementations.

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.

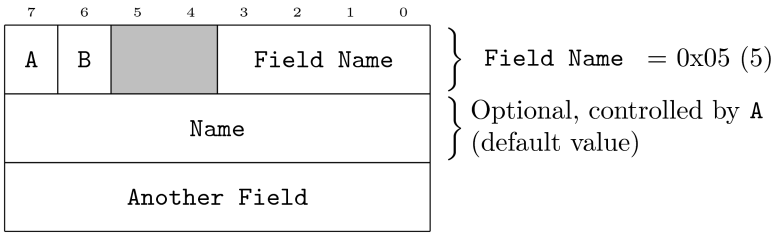
1.3.4. Fields

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. Overview

The Ticket Request Protocol (TRP) is used to communicate between a node and an authentication server (AS). As with other protocols in the DOF Protocol Stack, TRP may be relayed from nodes that are not an AS to a node that is an AS in a transparent manner. This is a form of proxy that is common in DOF systems. The end target of the communication is a domain, and the domain is identified in all TRP communication.

The purpose of communicating with a domain is usually to obtain keys that can be used to establish secure communication with another node (not the AS itself). This is the case with almost all of the TRP commands. Exceptions to this normal case are the Request Security Scopes and Resolve Credential commands, which do not distribute keys used to secure further communication. However, all communication with the domain is authenticated. This means that it is possible for the node sending the command to determine that the command was responded to by a domain and that the command and response were not modified in any way. The trust established depends on credentials provided by the node sending the command, and verification of the responder's knowledge of a shared secret that is associated with that credential.

TRP is not used directly between nodes that need to establish secure communication, but usually only by the server node. Nodes may communicate using TRP in some cases, but this usually part of a proxy relationship.

TRP is used in the following cases:

- By a server, when an inbound session requests a session key. (Request Session)
- By a node joining a secure group, to obtain the group's security state information. (Request KEK)
- By a node that knows about the presence of an AS and is acting as proxy. (all commands)
- By a node, to request security information about another node. (Request Security Scopes)
- By a node, to obtain a cryptographically random number. (Request Random)
- By a node, to establish credentials in a remote domain. (Request Remote Domain)
- By a node, to resolve credential information that is not already a shared key. (Resolve Credential)

In each of these cases, a node first builds a command. The command includes unique information identifying the command and information about the requesting node and the target domain. Receiving nodes either have information about the requested domain, are able to forward the command to another node, or send a Reject response.

An Authentication Server that contains information about the domain formulates a response to the command. The response is sent back to the sender of the command. In most cases, this will complete the command. The exception to this rule is the Resolve Credential command, which may require that state be kept and that the command be retried (providing additional information) and further related responses be sent. The details on this are discussed in the specific command sections.

Keys distributed by the AS are always limited to a scope that includes a set of nodes and duration. In other words, the information distributed can only be understood completely by a limited number of nodes (usually one or two) and is only valid for a specific period from the time of the request. After that period, a node again communicates with the AS to obtain new information. This process continues for as long as the nodes desire to communicate securely. This allows the domain to change information (or react to changed information) and to ensure that the results of these changes take place in a bounded time.

Note that the sense of a session key exists only between the two communicating nodes and not the AS. This ensures (for the most part) the stateless nature of the AS.

TRP is closely related to the protocols that exchange requests and tickets between non-AS nodes. This is because the information passed in TRP comes (at least partially) from these protocols, and the responses are sent back using the same protocol. The cryptography used between these nodes and the authentication server is also interrelated.

2.1. Operation State vs. Authentication Server State

It is important to understand the difference between operation state and authentication server state. A design goal of TRP is to minimize the use of operation state – and most operations require no operation state at all. However, since the goal of the authentication server is to centralize security information, there is no goal to minimize authentication server state. This means that TRP operations generally read information from the AS and respond, and that the operation is then complete. This also means that the majority of the operation state functionality provided by the DOF Protocol Stack is unused by TRP.

The sole exception to this is the Resolve Credential command. Depending on the credential type this command may require operation state. The general rule is that credential resolution follows the lifecycle of the DOF Protocol Stack. This means that operation state is used to tie the operation together over time as multiple (different) resolution stages occur.

However, even this general rule has an exception for domain discovery. In the case of domain discovery there may be multiple authentication servers responding. In this case there is no operation state kept.

2.2. Local and Remote Domains

One of the more complicated aspects of TRP is the management of "remote domains". In order to understand this concept a brief overview of a domain is required. For more information please refer to the *DOF Security Specification*.

A security domain is a collection of security information including authentication information, node identities and access control information. The access control information controls which nodes are allowed to communicate and in what ways.

In order for a node to communicate in a domain the domain must have a credential for the node. This generally means that the node must share a secret with the domain. However, there are topologies where a system may allow traffic from external domains to be routed through it. This is called a "remote" domain.

In this use case one of the two nodes knows that it must authenticate into a domain for which it does not have credentials, although it has credentials in some other domain. In order to obtain a shared key with the remote domain it must communicate first with the remote domain and then with the local domain (with which it does share credentials). At the end of this process it will have valid credentials in the remote domain. It will also have obtained information from the local domain about how to partition the traffic from the remote domain inside the local domain.

3. Specification

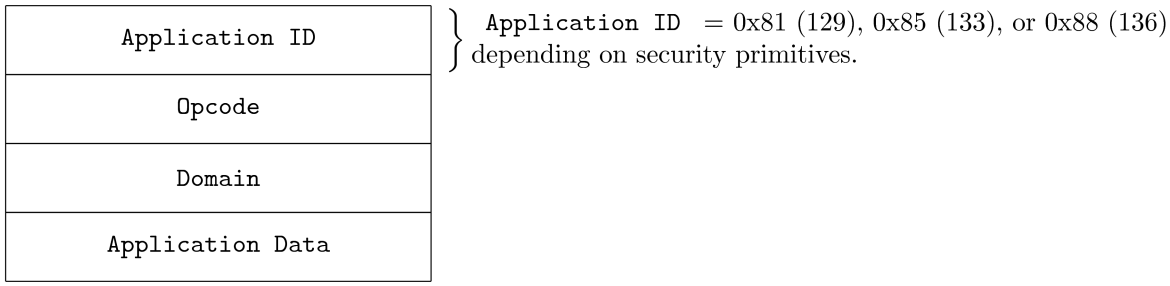
This section contains the specification of the Ticket Request Protocol.

3.1. Application Format

TRP uses an application identifier of 0x81 (129). When used to send commands, the following format is used.

⇄ **TRP Application Command Format** `dof-2008-5-pdu-1`

Instance of General Application Header`dof-2008-1-pdu-13`



`Application ID` **Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.Contains the application identifier for the enclosed data.

`Opcode` **One byte.** The value will indicate the particular operation.

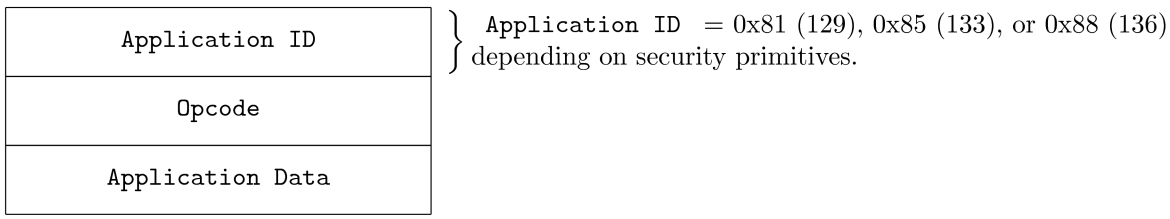
`Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The domain of the request.

`Application Data` **Variable length. Optional, controlled by** `Opcode`. Each **Opcode** will define the required **Application Data**.

When used to send responses, the following format is used.

⇄ **TRP Application Response Format** `dof-2008-5-pdu-2`

Instance of General Application Header`dof-2008-1-pdu-13`



`Application ID`	**Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.Contains the application identifier for the enclosed data.
`Opcode`	**One byte.** The value will indicate the particular operation.
`Application Data`	**Variable length. Optional, controlled by** `Opcode`. Each **Opcode** will define the required **Application Data**.

3.2. Security Primitives

This protocol version uses the following security primitives in place of the general primitives defined in the *DOF Security Specification* based on APPID:

APPID	ENC	HASH	MAC
0x81 (129)	AES	SHA-256	HMAC
0x85 (133)	Twofish	SHA-256	HMAC
0x88 (136)	SMS4	SHA-256	HMAC

3.3. Negotiation

Nodes request to use TRP on lossless 2-node sessions during DSP negotiation. Nodes that can accept TRP requests should acknowledge these requests. An AS always accepts a request to speak TRP. TRP is a client/server protocol, with either or both nodes being able to negotiate its use.

It is difficult for a server to know whether to request TRP on a new session or not. In general, it is not known whether an incoming request will use a domain that is unknown to the server. This determination is made by the implementation. If there is a chance that a valid inbound request would be for a domain that is not known to the server then the server should request TRP during DSP. A client determines whether to request TRP using the same logic. If the client may use credentials that require communicating with a domain that it cannot communicate with directly then TRP should be requested.

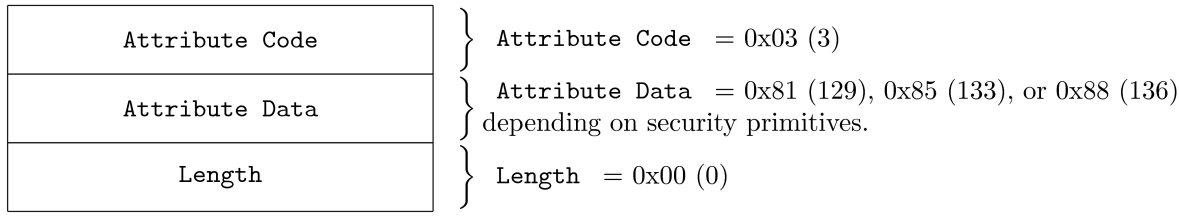
Nodes must allow TRP during authentication if it was negotiated in DSP. **`dof-2008-5-spec-1`**

This is allowed because TRP does not require security (encryption) as it authenticates responses in other ways.

There are no options for TRP negotiation. The following request is always used:

DSP Authentication Attribute **`dof-2008-5-pdu-3`**

Instance of Attribute/Value Pair`dof-2008-1-pdu-16`



`Attribute Code` **One byte.**

`Attribute Data` **Two bytes, MSB.**

`Length` **One byte.**

If a node allows a TRP request then it should either respond itself as an AS or proxy the request to another node.

3.4. Multicast Behavior

In order to allow multicast to be used with TRP the following rules are required:

- Failures to commands that are sent using transport multicast should not result in a Reject response.
- The first successful, validated response received to a request will be used, and future responses (if any should arrive) should be ignored.

Respond correctly to multicast commands. `dof-2008-5-spec-2`

This includes forwarding the request over other sessions that may be able to respond.

3.5. Authentication Server Discovery

In the case that the security domain identifier is known but the transport address of the AS is not known, AS discovery can be used. This case relies on the multicast behavior described above. Any of the commands described in this document may be used for discovery if the security domain identifier is known. Once a valid response is received, the transport address of the responding AS (or proxy) may be remembered for later use. However, since TRP commands and responses are not required to be secure themselves (at the transport layer, at least) it is possible for the location to be incorrect or forged. This means that if the location fails in the future then multicast can be used again.

In practice, there is little overhead to using multicast in this way.

3.6. Domain Discovery

Domain discovery is very different from AS discovery. In domain discovery, only a partial credential is known. In particular, the node identification and authentication information are known, but the security domain identifier is not known.

Domain discovery also uses multicast, but in this case, the security domain identifier will be the broadcast OID. Domain discovery is computationally much more expensive because each AS evaluates the credential to see if it is valid.

Only the Resolve Credential and Request Remote Domain commands may be used for domain discovery. See those commands for more information.

Domain discovery can also be expensive in terms of network traffic because proxy nodes may have to "fan out" a single command to multiple targets.

3.7. Routing

All TRP operations are listed as using session security. This means that operations passed on the wire may be encrypted, although there are no permissions required for TRP operations (commands or responses). A key distinction with TRP routing is that it does *not* follow the typical secure routing methods of determining compatible sessions – rather it uses entirely application-based routing logic.

INFO: As an application protocol, TRP is not subject to the typical domain and permission group identifier partitioning. Lower-level stack layers may still be subject to partitioning.

As mentioned earlier, TRP commands can be forwarded from a receiving node to another node that may be an AS or another proxy. This is a type of proxy behavior. This section discusses this behavior.



Keep routing paths the same for each command as much as possible. `dof-2008-5-spec-3`

Nodes that can proxy TRP commands must use the same route for each command as long as possible. The Resolve Credential command maintains state on the AS, and so it is important that it always be sent to the same AS. If the route is no longer available then a new route may be chosen. If the router knows that there is no state requirement for a given operation then it is free to use whatever route it wants.

Note that the requirement can be met by either keeping track of the route for each command, in which case different commands can use different routes, or it can be met by routing all commands for each domain to the same place.

The stability requirement for routing of TRP brings up the question of how a node determines whether a given route is valid. This is a difficult question, given that TRP cannot rely on secure communications in the typical sense and all communication with the AS can be authenticated using other means.

The following logic would then apply to determining whether a route is valid for a given domain:

- The implementation is configured with the route. This is a typical case in general, as the security configuration for a node would likely include the location of an appropriate AS.
- The route has been used and returned a valid response.
- The route has been used for other operations that require the domain exist with a valid response. For example, if a session is secure, the node was the initiator, and TRP was not redirected back to the node then it can be assumed that the responder had access to the domain used for the session. This is enough to allow further TRP commands for that domain to use the session as a route.

There are two easy ways to elicit a valid response from an AS. The first is to use the Request Random command. This command will return a response that can be validated. The downside to that command is that it requires credentials that are already resolved. The second way is to use the Resolve Credential command. This command also returns a response (or sequence of responses) that are ultimately validated.

Only forward to nodes that have been configured or validated. `dof-2008-5-spec-4`

Valid routes are those that have been used for a given domain and which have returned a validated TRP response, used the domain in other ways (like TEP), or are configured.

A result of the requirement to validate routes is that the proxy node must have credentials in the domain that is being routed (unless the route is configured). This means that the implementation knows the set of domains for which it has credentials, but not routes. This set will usually be very small, and so issuing a command as discussed above in each domain does not require much overhead.

TRP utilizes cryptography as described in the *DOF Security Specification*. A critical requirement for DOF cryptography is the use of hash algorithms of information that is passed between the nodes and the domain. In order to ensure equivalent hash results, the PDUs that are used must not change as they are forwarded through the system. This requirement also means that care must be taken when forwarding a command using a different protocol version.

Ensure that the PDUs used in TRP are forwarded without modification. `dof-2008-5-spec-5`

This specification applies to all PDUs used in TRP and PDUs that they use.

Note that the reliance on application configuration for routing may result in loops in the TRP routing paths. This can be determined by a node if it has multiple configured paths for the same domain. In this case the operation must be FLOODED instead of DIRECTED. In the typical case a DIRECTED operation may be used.

3.8. Session Requirements

TRP does not require a secure session. However, it is possible to send all TRP commands over an already secured session.

TRP may be used over both lossless and lossy sessions. In the case of lossy sessions, commands may need to be retried because of datagram loss.

3.9. Operation State Requirements

With one exception, TRP commands do not require persistent state on operations. This means that once a response (and any retransmissions) has been sent, that the operation state can be removed.

The exception is the Resolve Credential command. State is kept corresponding to the current stage of that operation until either the command expires or until it is cancelled.

3.10. Security Considerations

TRP is used in two cases: 1) during security negotiation in order to communicate with the AS, and 2) on established sessions, usually because of application configuration.

It is not possible to determine which case applies to a particular command, and this means that TRP commands are always allowed on a session. However, once a session is established then TRP must follow the security of the session.

To summarize, TRP commands are allowed in the following cases:

- All unsecured (established) sessions, including "None".
- All secured (established) sessions.
- During session security negotiation (which will be unsecured).

This results in a very interesting possibility, which is allowed, that an unsecured command results in a secure response.

3.11. Request Types

Communication with the authentication server usually results through maintaining a secure session, part of which involves permission validation. DOF specifications support two fundamentally different types of session: 2-node and n-node. 2-node encrypted sessions ensure that information is only available to the two nodes, where n-node sessions allow many nodes access to the information.

It is possible to restrict permission grants based on the two of use. In order to accomplish this, the authentication server needs to know the type of the request. In addition, information about the security level is known to the authentication server. In typical session requests this information is inherent in the request itself. A special case is the Request Security Scopes command, which passes the information directly.

4. Command Reference

The Ticket Request Protocol consists of messages sent between a requesting node and an authentication server or proxy. This section describes the commands and responses that make up the Ticket Request Protocol.

The *DOF Security Specification* introduces a common pseudo-language for security primitives. TRP can be used for the following statements:

AS?: (A), (B) $\Rightarrow$ A $\leftrightarrow$ B with perms: $\varnothing$, $\varnothing$, where a Request Session command is used.

AS?: (A), (G) $\Rightarrow$ A $\rightarrow$ G with perms: $\{P\}_A$, $\{P\}_{GA}$, where a Request KEK command is used.

A: (A, …), (B, …) # A # B with permissions: $\{P\}_A$, $\{P\}_B$; key: K, where a Request Session response is received.

A: (A, …), (G) # A # G with perms:$\{P\}_A$, $\{P\}_G$; session S_2; kek: K, where a Request KEK response is received.

A: (A), (B) # #, where a Reject response is received.

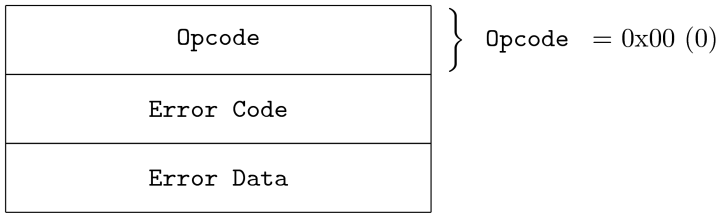
A: (A, …), (G) # #, where a Reject response is received.

4.1. Reject

Notifies a requesting node of a failure.
All TRP commands can result in this response.
Session: Lossy (2-node), Lossless
Addressing: Unicast
Security: Must match that of the corresponding command.
Permission (command): N/A
Permission (response): None

⇄ **Reject** `dof-2008-5-pdu-4`

RESPONSE
Instance of TRP Application Response Format`dof-2008-5-pdu-2`



`Opcode` **One byte.**

`Error Code` **One byte.** A code that indicates the condition that caused the request to fail.

`Error Data` **Variable bytes to end of PDU.** Data associated with the error.

4.1.1. Flows

The following flows are allowed for this response:

command() $\rightarrow$ Reject() : Failure.

4.1.2. Details

The Reject response is used to signal an error to a request. Depending on the type of error, the operation failure may be permanent or temporary. It is also possible that the result will change due to changes in the configuration of the domain.

Note that some of these errors may be returned by a proxy node without the request ever being delivered to an authentication server.

The following values are allowed for the Error Code:

Table 4.1. TRP Error Codes

Error Code	Short Description	Error Data	Long Description
1	Parse Error	*none*	The receiver was not able to parse the request. This is most likely due to a defect or configuration error. All commands can return this.
2	Access Denied	*none*	The AS could not authorize the request because one or more requested nodes are not enabled. All commands can return this.
3	Unknown Initiator	*none*	The initiator is not known to the AS. All commands can return this.
4	Unknown Responder	*none*	The responder is not known to the AS. See Request Session.
5	Unknown Domain	*none*	The domain requested for authentication is unknown to the receiver. See Request Remote Domain.
6	Service Unavailable	*none*	No response can be generated because the service is not available. The same request may succeed later. All commands can return this.

Error Code	Short Description	Error Data	Long Description
7	Mode Error	DSP Command/ Response Format	The requested security mode is not acceptable. The **Error Data** contains a DSP response that can provide the requestor with guidance on making a future request. The response will be either a Configuration Negative Acknowledge or a Configuration Reject based on the capabilities and configuration of the AS.
9	Initiator Not Group Member	*none*	The initiator is not a member of the requested group. See Request KEK.
10	Unsupported Identification Initiator	*none*	The identification mode requested by the initiator is not known. Identification is not possible. All commands can return this.
11	Unsupported Identification Responder	*none*	The identification mode requested by the responder is not known. Identification is not possible. See Request Session.
12	Invalid Identification Initiator	*none*	The identification mode requested by the initiator is known, but there is no associated private data. All commands can return this.
13	Invalid Identification Responder	*none*	The identification mode requested by the responder is known, but there is no associated private data. See Request Session.
14	Resolution Failed	*none*	An error occurred in resolution. This may indicate that the authentication information was invalid, for example, a bad

Error Code	Short Description	Error Data	Long Description
			password was used. See Resolve Credential.
15	Validation Failed	*none*	An error occurred in validation. This indicates that the authentication information was invalid. See Validate Credential.
127	Internal Error	*none*	An unexpected error occurred. More information should be available in the AS logs. All commands can return this.

Unregistered TRP Error Codes are not allowed. `dof-2008-5-spec-6`

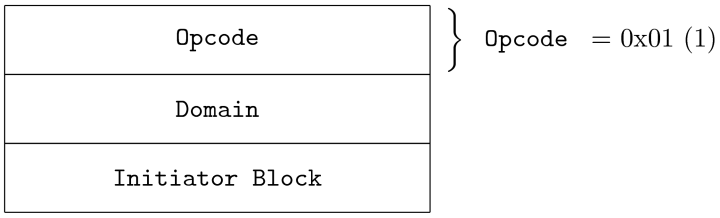
All unrecognised error codes or codes greater than 127 must be mapped to `Internal Error`.

4.2. Request KEK

Requests authentication into a secure group.
See Reject.
Session: None, Lossy (2-node), Lossy (n-node), Lossless
Addressing: Unicast, Multicast, Unicast
Security: See the section on Security Considerations, above.
Permission (command): None
Permission (response): None

Request KEK (command) `dof-2008-5-pdu-5`

COMMAND/DIRECTED or FLOODED
OPID
Instance of TRP Application Command Format`dof-2008-5-pdu-1`

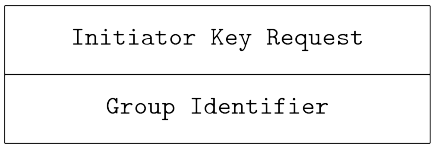


`Opcode` **One byte.** The value will indicate the particular operation.

`Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The domain of the request.

`Initiator Block` **Instance of** Request KEK Request Block`dof-2008-5-pdu-6`.This block represents client authentication information and permissions.

⇄ Request KEK Request Block `dof-2008-5-pdu-6`

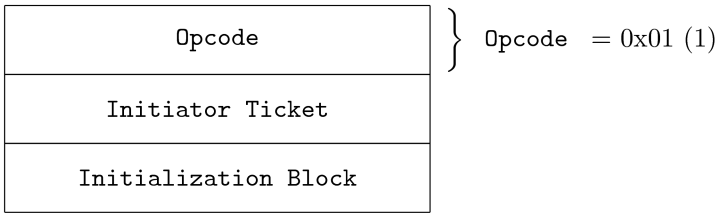


`Initiator Key Request` **Instance of** Key Request`dof-2008-16-pdu-13`.Initiator authentication information and permissions.

`Group Identifier` **Instance of** Security Node Identifier`dof-2008-16-pdu-1`.The identity of the group that the initiator wishes to join.

⇄ Request KEK (response) `dof-2008-5-pdu-7`

RESPONSE
OPID
Instance of TRP Application Response Format`dof-2008-5-pdu-2`

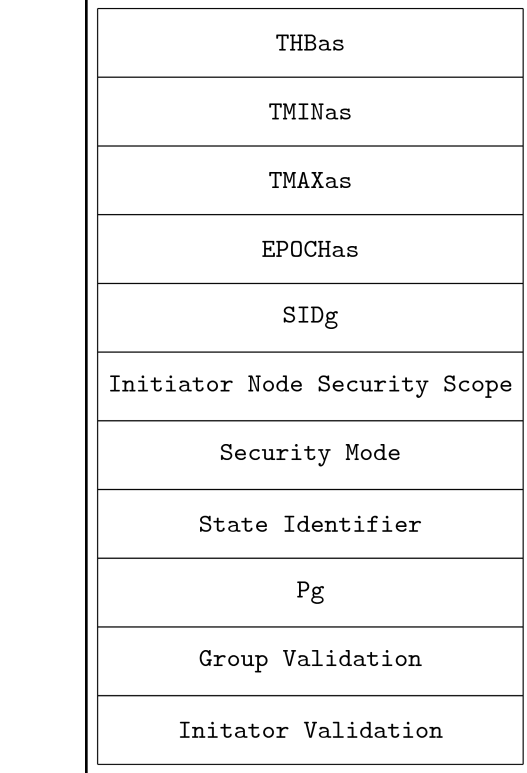


`Opcode` **One byte.** The value will indicate the particular operation.

`Initiator Ticket` **Instance of** Key Grant`dof-2008-16-pdu-14`.The initiator ticket information.

`Initialization Block` **Instance of** Request KEK Initialization Block`dof-2008-5-pdu-8`.Information about group parameters.

Request KEK Initialization Block `dof-2008-5-pdu-8`



`THBas` **One byte.** The heartbeat timing. The field contains seconds, with 1 (one) second units.

`TMINas` **One byte.** The minimum KEK timing. The field contains minutes, with 1 (one) minute units.

`TMAXas` **One byte.** The maximum KEK timing. The field contains minutes, with 8 (eight) minute units. This represents the expiration time of the permission grant and the associated KEK.

`EPOCHas` **Two bytes, MSB.** The epoch of the associated security parameters. This includes all parameters in this PDU *except* those specific to the initiator (**Initiator Node Security Scope** and **Initiator Validation**).

`SIDg` **Instance of** AS Assigned`dof-2009-1-pdu-38`.The SID assigned to the requestor for use in the secure group.

`Initiator Node Security Scope` **Instance of** Security Scope`dof-2008-16-pdu-6`.The initiator security scope.

`Security Mode` **Instance of** Security Mode of Operation Negotiation`dof-2008-16-pdu-11`.The security mode of the group. This is the same format as is usually put in a request,

	so it can be thought of as the AS requesting the node to use this security mode.
`State Identifier`	**Instance of** Compressed Unsigned 32-Bit`dof-2009-1-pdu-3`.The security state identifier that the node must use with this group.
`Pg`	**Instance of** Permission Request`dof-2008-16-pdu-8`.The effective group permission set as described in the *DOF Security Specification*.
`Group Validation`	**Instance of** Permission Validation`dof-2008-16-pdu-9`.This validation is for $\mathbf{P_G}$. Its count must match that of $\mathbf{P_G}$.
`Initator Validation`	**Instance of** Permission Validation`dof-2008-16-pdu-9`.This validation is for the permissions in the **Initialization Block** from the request.

4.2.1. Flows

Request KEK() $\rightarrow$ response() : Success.

Request KEK() $\rightarrow$ Reject() : Failure to join a group.

4.2.2. Details

This command is used by an initiating node to begin establishment of membership in a secure group. If successful then the node will obtain a KEK, which it can use to communicate with a group manager.

If the request is successful then a Request KEK response will be sent. This response includes the group security state information, validation of the permission request, and matching permission information about the group. It is an error if any of the timing parameters is zero, in which case the response must be ignored.

The permissions ($\mathbf{P_G}$) that are returned in the response require explanation. In the request, the node joining the group places all the permissions that it requests to interact with the group. However, there is no opportunity to negotiate individually with other nodes in the group. This means that the effective permissions of the group (including all members of the group) needs to be disclosed to the node as part of this request. The specifics of how this set is created are discussed in the *DOF Security Specification*.

When an authentication server receives this command, it determines whether a response should be sent. The following sequence of checks are done in order, generating the indicated responses.

- If the initiator's authentication mode is unsupported, send Reject(Unsupported Identification Initiator).
- If the AS storage is unavailable, send Reject(Service Unavailable).
- If the initiator information cannot be located, send Reject(Unknown Initiator).
- If the initiator is not enabled, send Reject (Access Denied).
- If the group information cannot be located, send Reject(Unknown Responder).

- If the group is not enabled, send Reject(Access Denied).
- If the initiator is not a group member, send Reject(Initiator not in Group).
- If the initiator's authentication mode is invalid, send Reject(Invalid Identification Initiator).
- If any other error occurs, send Reject(Internal Error).
- Finally, send a successful response.

4.2.3. Cryptography

This command uses the two-party authentication method described in the *DOF Security Specification*. The following blocks are used:

Table 4.2. Request KEK Cryptography Parameters

Parameter Name	Parameter Contents
I	Request KEK Request Block, from the command.
A	Request KEK Initialization Block, from the response.
K_I	The initiator's shared 256-bit private key with the authentication server.
K_D	KEK. This is the current group KEK.

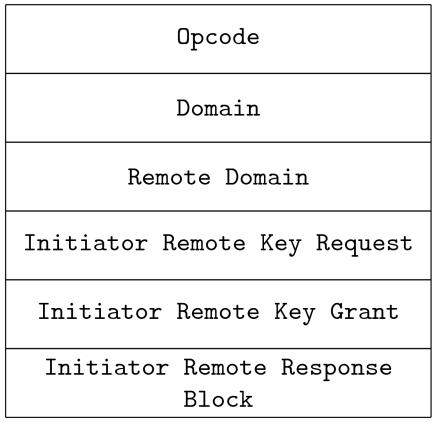
4.3. Request Local Domain

Requests that a remote domain request be refactored by the local domain of the initiator.
See Reject.
Session: None, Lossy (2-node), Lossy (n-node), Lossless
Addressing: Unicast, Multicast, Unicast
Security: See the section on Security Considerations, above.
Permission (command): None
Permission (response): None



Request Local Domain (command) `dof-2008-5-pdu-9`

COMMAND/DIRECTED or FLOODED
OPID
Instance of TRP Application Command Format`dof-2008-5-pdu-1`



`Opcode` **One byte.** The value will indicate the particular operation.

`Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The domain of the request.

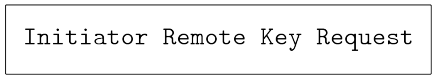
`Remote Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The remote domain of the request.

`Initiator Remote Key Request` **Instance of** Request Local Domain Request Block`dof-2008-5-pdu-10`.This block represents the initiator request to the remote domain and includes the local identity information.

`Initiator Remote Key Grant` **Instance of** Key Grant`dof-2008-16-pdu-14`.Information representing initiator authentication information.

`Initiator Remote Response Block` **Instance of** Request Remote Domain Response Block`dof-2008-5-pdu-20`.This block represents initiator authentication information. This needs to be the same as was received in the Request Remote Domain response.

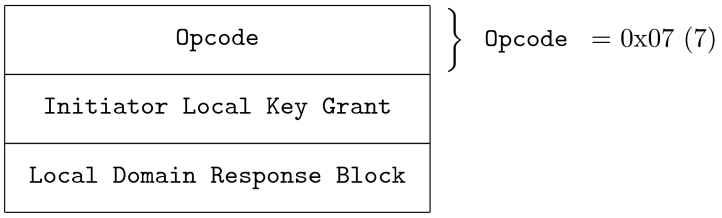
Request Local Domain Request Block `dof-2008-5-pdu-10`



`Initiator Remote Key Request` **Instance of** Request Remote Domain Request Block`dof-2008-5-pdu-17`.his block represents the initiator request to the remote domain, and needs to be the same as passed in the Request Remote Domain command. It identifies the local identity for authentication.

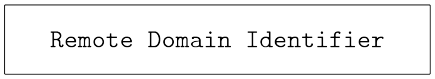
⇄ Request Local Domain (response) `dof-2008-5-pdu-11`

RESPONSE
OPID
Instance of TRP Application Response Format`dof-2008-5-pdu-2`



`Opcode`	**One byte.** The value will indicate the particular operation.
`Initiator Local Key Grant`	**Instance of** Key Grant`dof-2008-16-pdu-14`.This block represents the initiator key grant and validation.
`Local Domain Response Block`	**Instance of** Request Local Domain Response Block`dof-2008-5-pdu-12`.This block represents information from the local domain.

⇄ Request Local Domain Response Block `dof-2008-5-pdu-12`



`Remote Domain Identifier`	**Instance of** Compressed Unsigned 32-Bit`dof-2009-1-pdu-3`.This identifier is used in the local domain to identify traffic associated with the remote domain.

4.3.1. Flows

The following flows are allowed for this PDU. See the Stateless Command/Response Sequence Diagrams for examples.

Request Local Domain() $\rightarrow$ response() : Success.

Request Local Domain() $\rightarrow$ Reject() : Failure to establish a credential.

4.3.2. Details

This command is used by an initiating node when it has requested credentials in a remote domain for which it does not have a shared secret. In that case the Request Remote Domain response has returned a ticket that the initiator does not understand because it is signed with a shared domain key.

The initiator communicates with an authentication server that controls the local domain in which the node does have a shared secret. This command is used to request that the authentication server extract and validate the information and sign the initiator ticket with the initiator's shared secret.

If this is accomplished then the initiator is able to extract the shared session key, which in this case is the shared secret of the initiating node in the remote domain.

When an authentication server receives this command, it determines whether a response should be sent. The following sequence of checks need to be done in order, generating the indicated responses.

- If the initiator's authentication mode is unsupported, send Reject(Unsupported Identification Initiator).
- If the AS storage is unavailable, send Reject(Service Unavailable).
- If the remote domain cannot be located, send Reject(Unknown Domain).
- If the remote domain is not enabled, send Reject(Access Denied).
- If the initiator information cannot be located, send Reject(Unknown Initiator).
- If the initiator is not enabled, send Reject(Access Denied).
- If the initiator's authentication mode is invalid, send Reject(Invalid Identification Initiator).
- If the remote domain cannot be authenticated, send Reject(Access Denied).
- If any other error occurs, send Reject(Internal Error).
- Finally, send a successful response.

4.3.3. Cryptography

Note that the cryptography for this command occurs in two phases. In the first phase, the local domain is the recipient of the information from the remote domain, and uses the cryptography parameters specified in the Request Remote Domain command to extract K_D. In this phase the local domain authenticates the remote domain, including validating the response cryptographically. If this phase fails then the operation is rejected. Note that there are two domains in this command; the domain used in the hashing for this first phase is the remote domain of the command, *not* the target domain.

In the second phase, the local domain is the sender and uses the two-party authentication method described in the *DOF Security Specification*. Note that there are two domains in this command; the domain used in the hashing for this second phase is the target domain of the command, *not* the local domain.

The following blocks are used:

Table 4.3. Request Local Domain Cryptography Parameters

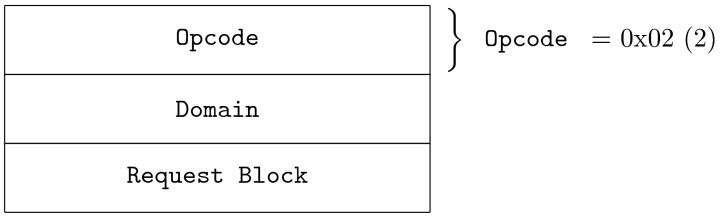
Parameter Name	Parameter Contents
I	Request Local Domain Request Block, from the command.
A	Request Local Domain Response Block, from the response.
K_I	The shared 256-bit secret between local domain and the local node.
K_D	The shared key between the remote domain and the initiator.

4.4. Request Random

Requests a random number suitable for use as a key.
See Reject.
Session: None, Lossy (2-node), Lossy (n-node), Lossless
Addressing: Unicast, Multicast, Unicast
Security: See the section on Security Considerations, above.
Permission (command): None
Permission (response): None

⇄ Request Random (command) `dof-2008-5-pdu-13`

COMMAND/DIRECTED or FLOODED
OPID
Instance of TRP Application Command Format`dof-2008-5-pdu-1`

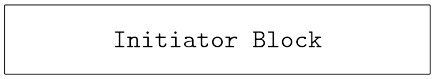


`Opcode` **One byte.** The value will indicate the particular operation.

`Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The domain of the request.

`Request Block` **Instance of** Request Random Request Block`dof-2008-5-pdu-14`.The request block.

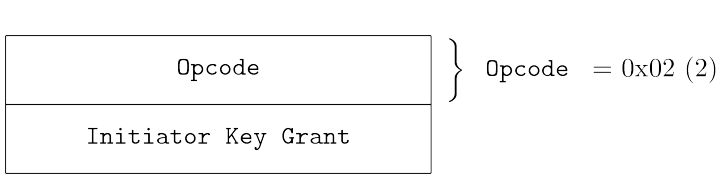
⇄ Request Random Request Block `dof-2008-5-pdu-14`



`Initiator Block` **Instance of** Key Request`dof-2008-16-pdu-13`.This block represents initiator authentication information.

⇄ Request Random (response) `dof-2008-5-pdu-15`

RESPONSE
OPID
Instance of TRP Application Response Format`dof-2008-5-pdu-2`



`Opcode`	**One byte.** The value will indicate the particular operation.
`Initiator Key Grant`	**Instance of** Key Grant`dof-2008-16-pdu-14`.This block authenticates the response and distributes the random number.

4.4.1. Flows

The following flows are allowed for this PDU. See the Stateless Command/Response Sequence Diagrams for examples.

Request Random() $\rightarrow$ response() : Success.

Request Random() $\rightarrow$ Reject() : Failure to get a random number.

4.4.2. Details

This command is used by an initiating node to obtain a cryptographically random number suitable to use as a session key or any other purpose. Many limited resource nodes have difficulty generating randomness, and this command provides a way that they can securely request a key from an AS. If successful then the node will obtain a random number which is known only to itself (and the AS, which will forget the assignment immediately).

If the request is successful then a Request Random response will be sent.

When an authentication server receives this command, it determines whether a response should be sent. The following sequence of checks need to be done in order, generating the indicated responses.

- If the initiator's authentication mode is unsupported, send Reject(Unsupported Identification Initiator).
- If the AS storage is unavailable, send Reject(Service Unavailable).
- If the initiator information cannot be located, send Reject(Unknown Initiator).
- If the initiator is not enabled, send Reject(Access Denied).
- If the initiator's authentication mode is invalid, send Reject(Invalid Identification Initiator).
- If any other error occurs, send Reject(Internal Error).
- Finally, send a successful response.

4.4.3. Cryptography

This command uses the two-party authentication method described in the *DOF Security Specification*. The following blocks are used:

Table 4.4. Request Random Cryptography Parameters

Parameter Name	Parameter Contents
I	Request Random Request Block, from the command.
A	Not used, treated as empty.
K_I	The initiator's shared 256-bit private key with the authentication server.
K_D	A 256-bit number suitable for use by the initiator as a session key.

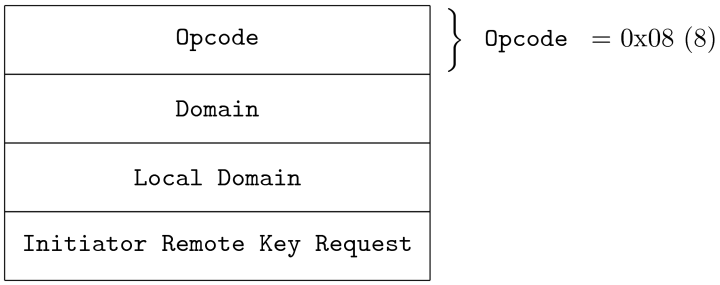
4.5. Request Remote Domain

Requests credentials from a remote domain (one in which the initiator does not have a shared key).
See Reject.
Session: None, Lossy (2-node), Lossy (n-node), Lossless
Addressing: Unicast, Multicast, Unicast
Security: See the section on Security Considerations, above.
Permission (command): None
Permission (response): None



Request Remote Domain (command) `dof-2008-5-pdu-16`

COMMAND/DIRECTED or FLOODED
OPID
Instance of TRP Application Command Format`dof-2008-5-pdu-1`



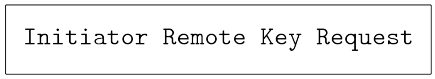
`Opcode` **One byte.** The value will indicate the particular operation.

`Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The domain of the request. In the case of associated domain discovery this can be the Broadcast OID, although attributes are not allowed.

`Local Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The local domain of the request.

`Initiator Remote Key Request`	**Instance of** Request Remote Domain Request Block`dof-2008-5-pdu-17`.Information representing the initiator request in the remote domain.

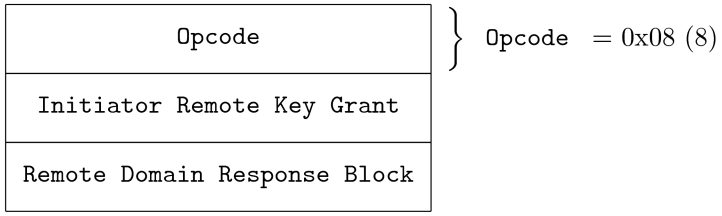
⇄ **Request Remote Domain Request Block `dof-2008-5-pdu-17`**



`Initiator Remote Key Request`	**Instance of** Key Request`dof-2008-16-pdu-13`.Information representing the initiator request in the remote domain.

⇄ **Request Remote Domain (response) `dof-2008-5-pdu-18`**

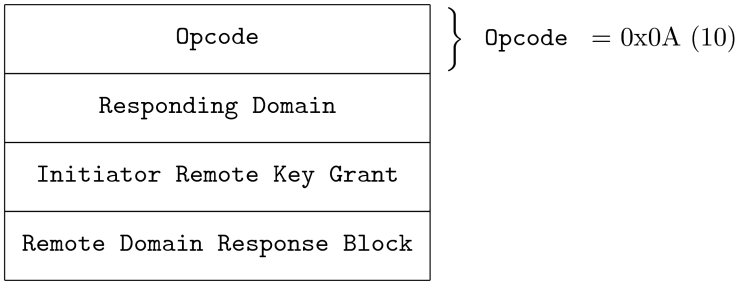
RESPONSE
Instance of TRP Application Response Format`dof-2008-5-pdu-2`



`Opcode`	**One byte.** The value will indicate the particular operation.
`Initiator Remote Key Grant`	**Instance of** Key Grant`dof-2008-16-pdu-14`.Information representing initiator authentication information.
`Remote Domain Response Block`	**Instance of** Request Remote Domain Response Block`dof-2008-5-pdu-20`.Remote domain information.

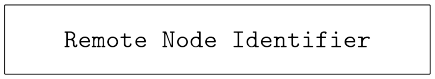
⇄ **Request Discovered Remote Domain (response) `dof-2008-5-pdu-19`**

RESPONSE
Instance of TRP Application Response Format`dof-2008-5-pdu-2`



`Opcode`	**One byte.** The value will indicate the particular operation.
`Responding Domain`	**Instance of** Security Domain`dof-2008-16-pdu-5`.The domain identifier of the responding domain.
`Initiator Remote Key Grant`	**Instance of** Key Grant`dof-2008-16-pdu-14`.Information representing initiator authentication information.
`Remote Domain Response Block`	**Instance of** Request Remote Domain Response Block`dof-2008-5-pdu-20`.Remote domain information.

⇄ **Request Remote Domain Response Block `dof-2008-5-pdu-20`**



`Remote Node Identifier`	**Instance of** Security Node Identifier`dof-2008-16-pdu-1`.The identifier that the node should use in the remote domain.

4.5.1. Flows

The following flows are allowed for this PDU. See the Stateless Command/Response Sequence Diagrams for examples.

Request Remote Domain() $\rightarrow$ response() : Success.

Request Remote Domain() $\rightarrow$ Request Discovered Remote Domain() : Success.

Request Remote Domain() $\rightarrow$ Reject() : Failure to establish credentials in the remote domain.

4.5.2. Details

This command begins the process of obtaining shared-key credentials for a remote domain, meaning a domain with which the initiating node does not share a secret. In order to determine the correct credentials, the remote domain distributes the shared-key secret to the initiating node through a local domain. The remote domain and local domain already share a secret between themselves, and the local domain shares a secret with the initiating node.

The resulting credential in the remote domain can be used to authenticate directly. The domain of the resulting credential is that of the remote domain, the security node identifier is determined by the remote domain and passed to the initiator as the **Remote Node Identifier**. The distributed secret (obtained from the remote domain through the local domain) corresponds to a shared-key credential type in the remote domain.

The **Initiator Remote Key Grant** returned cannot be understood by the initiating node because it uses shared key of the domains. It is forwarded to the local domain using the Request Local Domain command in order to be of use to the initiating node.

Note that in the process of this exchange, the local domain will obtain knowledge of the shared secret between the remote domain and the initiating node. This is acceptable as there is already a high degree of trust between the remote and local domains.

Also note that a new node identifier is returned from the remote domain to the initiator. The key that is distributed to the initiating node is associated with this new node identifier in the remote domain. The distribution of a new identifier resolves issues with aliasing of node identifiers between the local and remote domain. When the node authenticates into the remote domain (which is the goal of the entire process), then it will use the remote domain identifier, the remote node identifier, and the distributed key.

This command may be used even if the security domain is not known. In this case, the **Domain** will be the broadcast OID. Each AS receiving such a command should evaluate the credential and see if it may be valid. If it may be, then the AS should send a response that includes the responding security domain identifier. The determination that a response should be sent includes the fact that the credential is understood and that a relationship exists with the indicated **Local Domain**.

When an authentication server receives this command, it determines whether a response should be sent. The following sequence of checks need to be done in order, generating the indicated responses.

- If the local node’s authentication mode is unsupported, send Reject(Unsupported Identification Initiator).
- If the AS storage is unavailable, send Reject(Service Unavailable).
- If the local domain cannot be located, send Reject(Unknown Domain).
- If the local domain is not enabled, send Reject(Access Denied).
- If there is no local node information for the domain, send Reject(Unknown Initiator).
- If the local node is not enabled, send Reject(Access Denied).
- If the initiator’s authentication mode is invalid, send Reject(Invalid Identification Initiator).
- If any other error occurs, send Reject(Internal Error).
- Finally, send a successful response.

4.5.3. Cryptography

This command uses the two-party authentication method described in the *DOF Security Specification*. Note that there are two domains in this command; the domain used in the hashing is the target domain of the command, *not* the local domain.

The following blocks are used:

Table 4.5. Request Remote Domain Cryptography Parameters

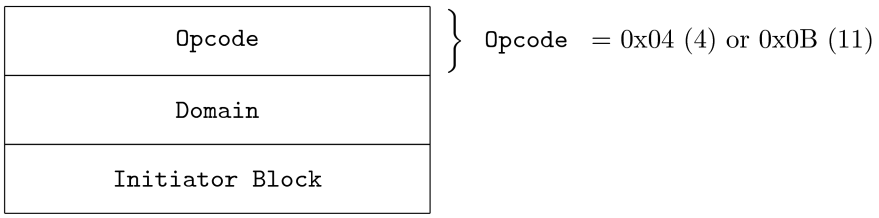
Parameter Name	Parameter Contents
I	Request Remote Domain Request Block, from the command.
A	Request Remote Domain Response Block, from the response.
K_I	The shared 256-bit secret between remote and local domains.
K_D	The shared key between the remote domain and the initiator.

4.6. Request Security Scopes

Requests security scope information on a specified node.
See Reject.
Session: None, Lossy (2-node), Lossy (n-node), Lossless
Addressing: Unicast, Multicast, Unicast
Security: See the section on Security Considerations, above.
Permission (command): None
Permission (response): None

Request Security Scopes (command) `dof-2008-5-pdu-21`

COMMAND/DIRECTED or FLOODED
OPID
Instance of TRP Application Command Format`dof-2008-5-pdu-1`

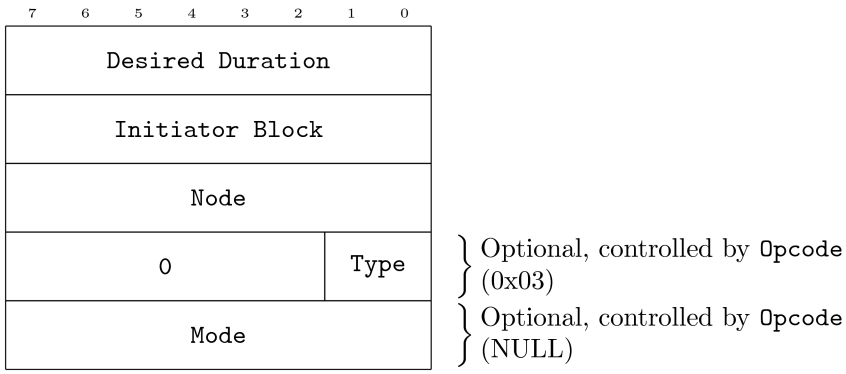


`Opcode` **One byte.** The value will indicate the particular operation.

`Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The domain of the request.

`Initiator Block` **Instance of** Request Security Scopes Request Block`dof-2008-5-pdu-22`.Information representing initiator authentication information.

Request Security Scopes Request Block `dof-2008-5-pdu-22`



`Desired Duration` **One byte.** The desired duration of the permission grant. The field is represented in minutes, with eight minute units. This represents the desired expiration time for the associated permission grants.

`Initiator Block` **Instance of** Key Request`dof-2008-16-pdu-13`.Information representing requestor authentication information.

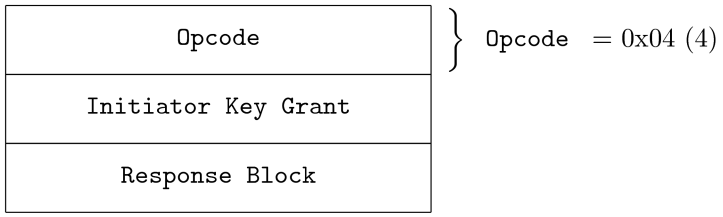
`Node` **Instance of** Security Node Identifier`dof-2008-16-pdu-1`.The node identifier to return security scopes for.

`Type` **Two bits.** Present if **Opcode** is 0x0B. Indicates the type of permission grants that should be considered. A value of 0x01 indicates 2-node, 0x02 indicates n-node, 0x03 indicates both 2-node and n-node. The value of 0x00 is invalid.

`Mode` **Instance of** Attribute/Value Pair`dof-2008-1-pdu-16`.Present if **Opcode** is 0x0B. The mode associated with the request. This allows the request to be evaluated in the context of the mode, allowing the consideration of how the operations granted will be secured.

Request Security Scopes (response) `dof-2008-5-pdu-23`

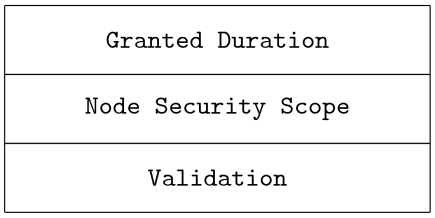
RESPONSE
OPID
Instance of TRP Application Response Format`dof-2008-5-pdu-2`



`Opcode` **One byte.** The value will indicate the particular operation.

`Initiator Key Grant`	**Instance of** Key Grant`dof-2008-16-pdu-14`.Information that authenticates the response.
`Response Block`	**Instance of** Request Security Scopes Initialization Block`dof-2008-5-pdu-24`.Response information.

⇄ **Request Security Scopes Initialization Block `dof-2008-5-pdu-24`**



`Granted Duration`	**One byte.** The granted duration of the permission grant. The field is represented in minutes, with eight minute units. The granted duration can be less than, equal to or greater than the requested duration. This represents the actual expiration time for the granted permissions.
`Node Security Scope`	**Instance of** Security Scope`dof-2008-16-pdu-6`.The scope of the requested node.
`Validation`	**Instance of** Permission Validation`dof-2008-16-pdu-9`.The validation of the permissions requested in the **Initiator Block** for the **Node**.

4.6.1. Flows

The following flows are allowed for this PDU. See the Stateless Command/Response Sequence Diagrams for examples.

Request Security Scopes() $\rightarrow$ response() : Success.

Request Security Scopes () $\rightarrow$ Reject() : Failure to get the security scopes.

4.6.2. Details

This command is used by an initiating node to obtain the security scopes that are assigned in its domain to another node. Note in particular that the permissions in the request are validated against *only* the requested node's granted permissions, and the permissions of the initiator have no effect on the result.

Note that this command uses two different opcodes, although the response uses only one. The second command opcode includes different fields, while the first uses default values.

When an authentication server receives this command, it determines whether a response should be sent. The following sequence of checks needs to be done in order, generating the indicated responses.

- If the initiator's authentication mode is unsupported, send Reject(Unsupported Identification Initiator).

- If the AS storage is unavailable, send Reject(Service Unavailable).
- If the initiator information cannot be located, send Reject(Unknown Initiator).
- If the initiator is not enabled, send Reject(Access Denied).
- If the requested node information cannot be located, send Reject(Unknown Responder).
- If the requested node is not enabled, send Reject(Access Denied).
- If the initiator's authentication mode is invalid, send Reject(Invalid Identification Initiator).
- If any other error occurs, send Reject(Internal Error).
- Finally, send a successful response.

4.6.3. Cryptography

This command uses the two-party authentication method described in the *DOF Security Specification*. The following blocks are used:

Table 4.6. Request Security Scopes Cryptography Parameters

Parameter Name	Parameter Contents
I	Request Security Scopes Request Block, from the command.
A	Request Security Scopes Initialization Block, from the response.
K_I	The initiator's shared 256-bit private key with the authentication server.
K_D	Ignored. Used to validate the response only.

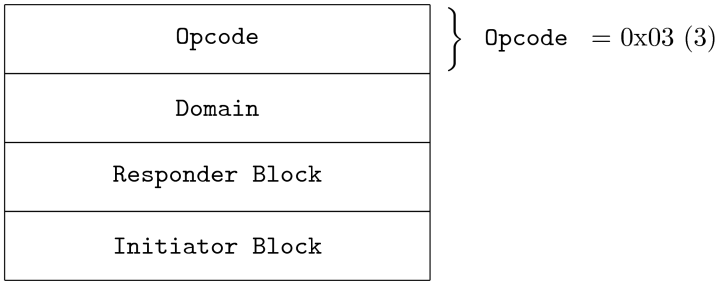
4.7. Request Session

Requests authentication and authorization of a secure session.
See Reject.
Session: None, Lossy (2-node), Lossy (n-node), Lossless
Addressing: Unicast, Multicast, Unicast
Security: See the section on Security Considerations, above.
Permission (command): None
Permission (response): None



Request Session (command) `dof-2008-5-pdu-25`

COMMAND/DIRECTED or FLOODED
OPID
Instance of TRP Application Command Format`dof-2008-5-pdu-1`



`Opcode` **One byte.** The value will indicate the particular operation.

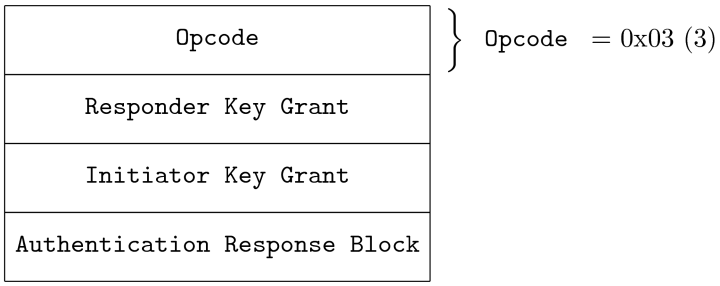
`Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The domain of the request.

`Responder Block` **Instance of** Session Responder Block`dof-2008-16-pdu-16`.Responder authentication information.

`Initiator Block` **Instance of** Session Initiator Block`dof-2008-16-pdu-15`.Initiator authentication information.

Request Session (response) `dof-2008-5-pdu-26`

RESPONSE
OPID
Instance of TRP Application Response Format`dof-2008-5-pdu-2`



`Opcode` **One byte.** The value will indicate the particular operation.

`Responder Key Grant` **Instance of** Key Grant`dof-2008-16-pdu-14`.Information that authenticates the session and grants a key to the responder.

`Initiator Key Grant` **Instance of** Key Grant`dof-2008-16-pdu-14`.Information that authenticates the session and grants a key to the initiator.

`Authentication Response Block` **Instance of** Authentication Response Block`dof-2008-16-pdu-17`.Information about the session.

4.7.1. Flows

The following flows are allowed for this PDU. See the Stateless Command/Response Sequence Diagrams for examples.

Request Session() $\rightarrow$ response() : Success.

Request Session() $\rightarrow$ Reject() : Failure to establish a session.

4.7.2. Details

This command is used by a node as part of establishing of a session with another node. If successful then they will each obtain a session key K_D and will have authenticated the node that is being communicated with as well as the permissions to enforce.

Note that there are two permission blocks in this TRP request. The first is in the initiator block, and the second in the responder block. The combination of these two permission blocks defines the permissions in the session. Permissions in each block are from the point of view of the supplier (the responder's permissions are from its perspective).

If the request is successful then a Request Session response will be sent.

The response includes security state information, including duration. This duration serves two purposes. First, due to the inevitable degradation of the security of an encrypted link over time, the cryptographic session keys that are received through this protocol cannot be granted indefinitely. Second, it is advantageous for changes in the AS to be guaranteed to take effect in some maximum delay. In other words, if an administrator removed some permission from a user in the AS, the administrator could be sure that after the maximum allowed expiration has elapsed, the user will no longer be able to use the resources controlled by those permissions.

A ticket is always output for both the initiator and responder.

The response also includes the authenticated initiator security identifier. More information on the use of the security identifier can be found in the *DOF Security Specification*.

When an authentication server receives this command, it determines whether a response should be sent. The following sequence of checks need to be done in order, generating the indicated responses.

- If the responder's authentication mode is unsupported, send Reject(Unsupported Identification Responder).
- If the initiator's authentication mode is unsupported, send Reject(Unsupported Identification Initiator).
- If the AS storage is unavailable, send Reject(Service Unavailable).
- If the requested security mode is not acceptable, return Reject(Bad Mode).
- If the initiator information cannot be located, send Reject(Unknown Initiator).
- If the initiator is not enabled, send Reject(Access Denied).
- If the responder information cannot be located, send Reject(Unknown Responder).
- If the responder is not enabled, send Reject(Access Denied).
- If the initiator's authentication mode is invalid, send Reject(Invalid Identification Initiator).

- If the responder's authentication mode is invalid, send Reject(Invalid Identification Responder).
- If any other error occurs, send Reject(Internal Error).
- Finally, send a successful response.

4.7.3. Cryptography

This command uses the three-party authentication method described in the *DOF Security Specification*. The following blocks are used:

Table 4.7. Request Session Cryptography Parameters

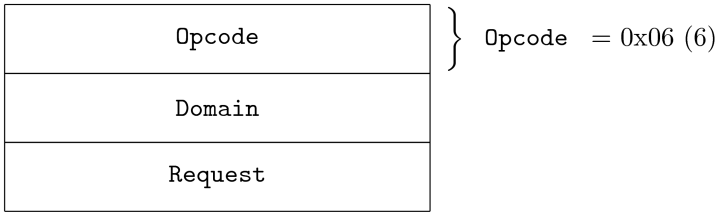
Parameter Name	Parameter Contents
I	Session Initiator Block, from the command.
R	Session Responder Block, from the command.
A	Authentication Response Block, from the response.
K_I	The initiator's shared 256-bit private key with the authentication server.
K_R	The responder's shared 256-bit private key with the authentication server.
K_D	The session key that should be used between initiator and responder.

4.8. Resolve Credential

Resolves a credential in cases where multiple data exchanges are required.
Also used for domain discovery.
See Reject.
Session: None, Lossy (2-node), Lossy (n-node), Lossless
Addressing: Unicast, Multicast, Unicast
Security: See the section on Security Considerations, above.
Permission (command): None
Permission (response): None

⇄ Resolve Credential (command) `dof-2008-5-pdu-27`

COMMAND/DIRECTED or FLOODED
OPID
Instance of TRP Application Command Format`dof-2008-5-pdu-1`



`Opcode` **One byte.** The value will indicate the particular operation.

`Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The domain of the request. In the case of domain discovery this may the broadcast OID.

`Request` **Instance of** Identity Resolution`dof-2008-16-pdu-3`.Data determined by the type of credential being resolved and the stage.

⇄ Resolve Credential (response) `dof-2008-5-pdu-28`

RESPONSE
OPID
Instance of TRP Application Response Format`dof-2008-5-pdu-2`

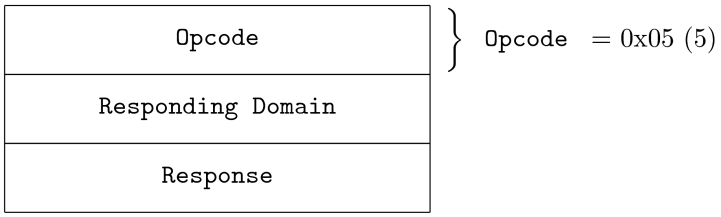


`Opcode` **One byte.** The value will indicate the particular operation.

`Response` **Instance of** Identity Resolution`dof-2008-16-pdu-3`.Data determined by the type of credential and the stage.

⇄ Resolve Discovered Credential (response) `dof-2008-5-pdu-29`

RESPONSE
OPID
Instance of TRP Application Response Format`dof-2008-5-pdu-2`



`Opcode` **One byte.** The value will indicate the particular operation.

`Responding Domain` **Instance of** Security Domain`dof-2008-16-pdu-5`.The domain identifier of the responding domain.

`Response` **Instance of** Identity Resolution`dof-2008-16-pdu-3`.Data determined by the type of credential and the stage.

4.8.1. Flows

The following flows are allowed for this PDU. See the Stateful Multiple Command/Response Sequence Diagrams for examples.

Resolve Credential() $\rightarrow$ response() : Success.

Resolve Credential() $\rightarrow$ Resolve Discovered Credential() : Success.

Resolve Credential() $\rightarrow$ Reject() : Failure to resolve a credential.

4.8.2. Details

Certain types of credentials do not immediately result in knowledge of a shared key that can be used for authentication. For example, a fingerprint is a valid authenticator, but is not a key. In these cases, multiple flows of information between the client and the authentication server may be required to determine a shared key that can be used.

As a further example, certain password authentication methods may start with the knowledge of a password, but may use numerical methods to determine a shared key based on the password. These cases pose problems for the authentication server, which is designed to be stateless.

In these cases, this command can be used. It serves the purpose of providing shared state (through the operation lifecycle) and allows the client and the authentication server to determine (or exchange) a shared secret that can then be used to authenticate the client node to other nodes.

The identity resolution request includes a stage that represents the status of the credential resolution. Its specific use and defined values are defined by each identification mode. The data (for both the command and response) are also defined by the identification mode. Its value must be greater than zero when used in this command.

This command may be used on lossy transports. In this case, the command and response may require retransmission to ensure they arrive.

This command may be retried. In each retry, the **Request** data may change. In all these cases, an appropriate response should be generated.

In addition, the command needs to ensure that correct responses are generated even if the requests are received out-of-order. Depending on the identification mode, this may require caching intermediate state data so that appropriate responses can always be sent. However, once a stage is received, requests that use a higher stage can be safely ignored.

This command may be used even if the security domain is not known. In this case, the **Domain** will be the broadcast OID. Each AS receiving such a command should evaluate the credential and see if it may be valid. If it may be, then the AS should send a response that includes the responding security domain identifier. However, in this case the lifecycle of the resolution is assumed to be terminated. If a single exchange of information is enough to complete resolution (indicating an initial stage 1 request), then no persistent state is required, and the sender has knowledge of the shared key. Otherwise, the responder should not keep state and the sender should not use the state to continue the resolution. Instead it should use the discovered domain from the response to initiate a new resolution request that does not use the broadcast OID. This process can be started in parallel with as many domains as are discovered.

Note that each credential type must ensure that the domain is authenticated as part of the credential resolution.

When an authentication server receives this command, it determines whether a response should be sent. The following sequence of checks need to be done in order, generating the indicated responses.

- If the initiator's authentication mode is unsupported, send Reject(Unsupported Identification Initiator).
- If the AS storage is unavailable, send Reject(Service Unavailable).
- If the initiator information cannot be located, send Reject(Unknown Initiator).
- If the resolution fails, send Reject(Resolution Failed).
- If any other error occurs, send Reject(Internal Error).
- Finally, send a successful response.

4.8.3. Routing

The use of the broadcast OID in this command raises the specific routing rules that apply in that case. In general a receiving node should forward these requests (keeping the broadcast OID) to each domain for which it has a validated route. If it knows of multiple routes to the same domain then it should only forward over one. Note that in the case of broadcast that there is no persistent state, and so the stability of the route does not matter.

Note that in this case the forwarding node should ignore responses from domain that are not validated over that route.

4.9. Validate Credential

Validates that the shared key associated with a credential is correct.
See Reject.
Session: None, Lossy (2-node), Lossy (n-node), Lossless
Addressing: Unicast, Multicast, Unicast
Security: See the section on Security Considerations, above.
Permission (command): None
Permission (response): None

⇄ **Validate Credential (command)** `dof-2008-5-pdu-30`

COMMAND/DIRECTED or FLOODED
OPID
Instance of TRP Application Command Format`dof-2008-5-pdu-1`

Opcode
Domain
Identity
VALIDa

} Opcode $= 0\text{x}09\ (9)$

`Opcode`	**One byte.** The value will indicate the particular operation.
`Domain`	**Instance of** Security Domain`dof-2008-16-pdu-5`.The domain of the request.
`Identity`	**Instance of** Base Credential Format`dof-2008-16-pdu-2`.The identity associated with the shared key to validate.
`VALIDa`	**Forty bytes.** Cryptographic information described below.

⇄ **Validate Credential (response)** **`dof-2008-5-pdu-31`**

RESPONSE
OPID
Instance of TRP Application Response Format`dof-2008-5-pdu-2`



`Opcode`	**One byte.** The value will indicate the particular operation.
`VALIDb`	**Variable length.** Cryptographic information described below.

4.9.1. Flows

The following flows are allowed for this PDU. See the Stateful Multiple Command/Response Sequence Diagrams for examples.

Validate Credential() $\rightarrow$ response() : Success.

Validate Credential() $\rightarrow$ Reject() : Failure to validate a credential.

4.9.2. Details

It can be beneficial to determine if a node's understanding of a shared key is correct. For example, after resolution the "temporary" shared key associated with the identity may change. Resolving the credential again will result in the new shared key being known, and validation allows the client to know if resolution is required.

This command is also used to obtain a source identifier from an AS based on a credential.

This command may be used on lossy transports. In this case, the command and response may require retransmission to ensure they arrive.

When an authentication server receives this command, it determines whether a response should be sent. The following sequence of checks need to be done in order, generating the indicated responses.

- If the initiator's authentication mode is unsupported, send Reject(Unsupported Identification Initiator).
- If the AS storage is unavailable, send Reject(Service Unavailable).

- If the initiator information cannot be located, send Reject(Unknown Initiator).
- If $VALID_A$ does not match, send Reject(Validation Failed).
- If any other error occurs, send Reject(Internal Error).
- Finally, send a successful response.

4.9.3. Shared-key Cryptography

The validation of shared keys is requires that 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.
SID	A SID, assigned by the AS, that is generated for **Identity**.
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

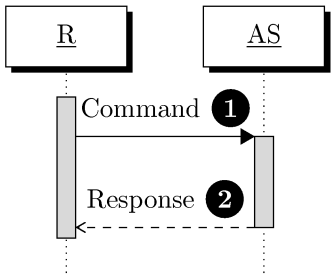
$VALID_B = msg_B \,||\, SID \,||\, MAC[\, K_{SHARED} \,](\, msg_A \,||\, msg_B \,||\, SID \,||\, D \,)$

5. Sequence Diagrams

All TRP commands fall into two categories: stateless command/response and stateful multiple command/response. The following sections show sequence diagrams for the two categories.

5.1. Stateless Command/Response

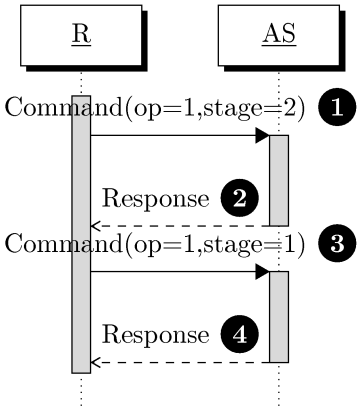
Figure 5.1. Stateless Command/Response



1. Command is sent from R to AS.
2. Response is sent from AS to R immediately.

5.2. Stateful Multiple Command/Response

Figure 5.2. Stateful Command/Response

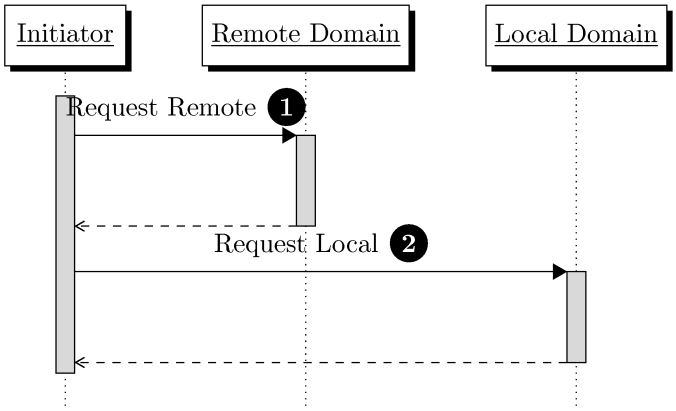


1. Command with operation identifier 1 is sent to the AS. This operation is at stage 2.
2. The AS responds to operation identifier 1, preserving state as the operation is not complete.
3. The command is retried (same operation identifier) but with stage 1.

4. The AS responds to the operation. The state may be preserved still to satisfy retries.

5.3. Local/Remote Domain

Figure 5.3. Stateful Command/Response

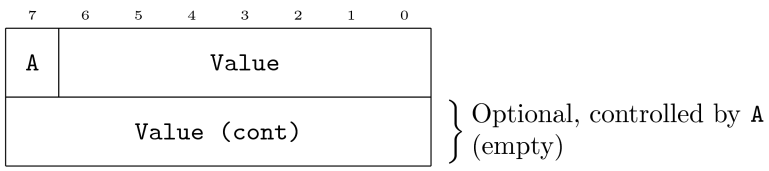


1. A request is made to the remote domain.
2. Based on the response, the request is made to the local domain.

6. External PDU Reference

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

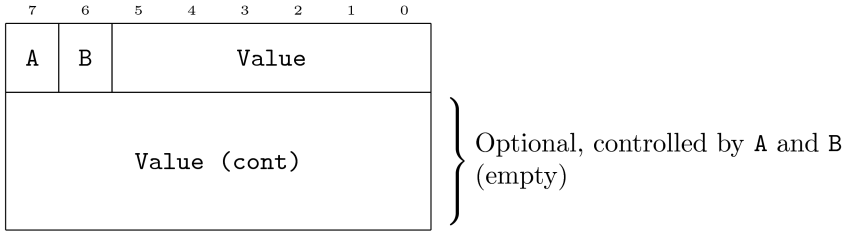
Compressed Unsigned 16-Bit `dof-2009-1-pdu-1`



`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`

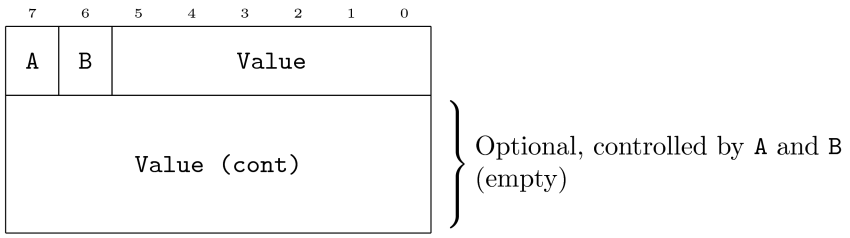


`A` **One bit.** Controls the size of `Value` and the presence of the `B` field.

`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`

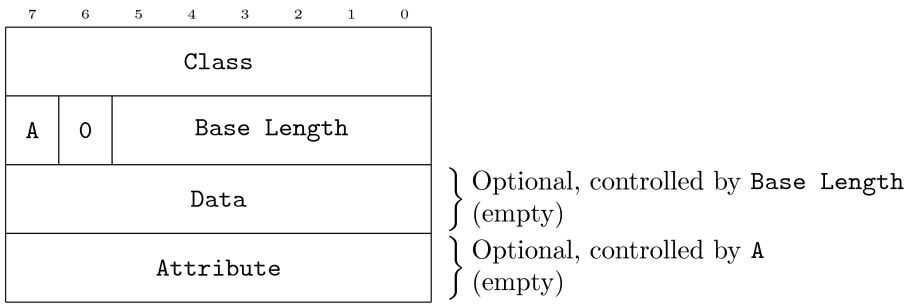


`A` **One bit.** Controls the size of `Value` and the presence of the `B` field.

`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.

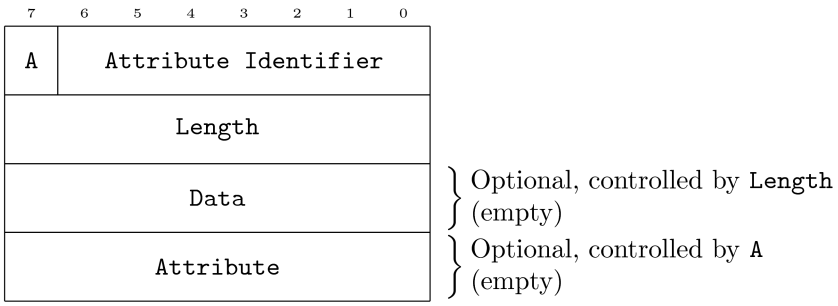
`A` **One bit.** Controls the presence of `Attribute`.

`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.

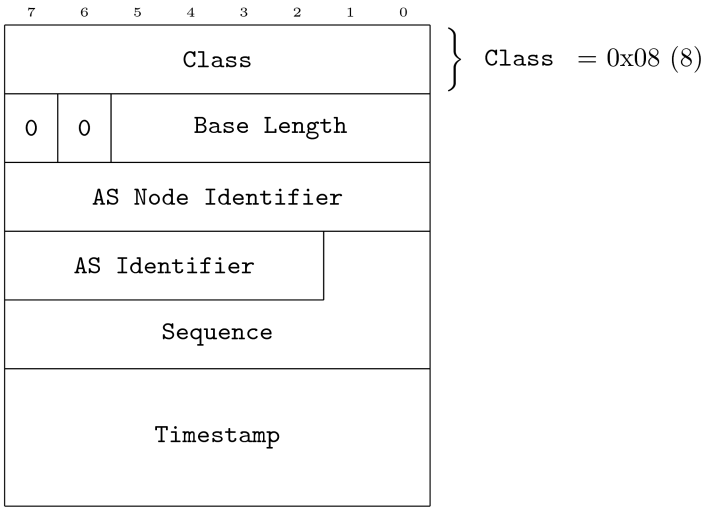
Object Identifier Attribute `dof-2009-1-pdu-5`



`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.

AS Assigned `dof-2009-1-pdu-38`

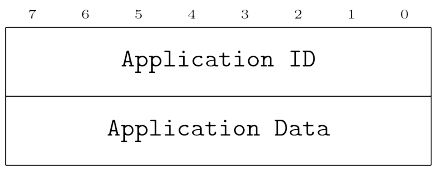
Instance of Object Identifier (OID)`dof-2009-1-pdu-4`



`Class`	**Instance of** Compressed Unsigned 32-Bit`dof-2009-1-pdu-3`.The OID class, registered with ODP-TSC.
`Base Length`	**Six bits.** This contains the sum of the `AS Node Identifier`, `AS Identifier`, `Sequence`, and `Timestamp` fields and is at least 0x07 (7) with a maximum length of 0x0A (10).
`AS Node Identifier`	**One to four bytes.** Contains the node identifier assigned by the AS to the credentials used to generate the OID. The format must not have leading zeroes, although the minimum size allows a single zero byte. The number of bytes in this field will be equal to `Base Length` minus six.
`AS Identifier`	**Six bits.** The identifier of the AS front end that is generating the OID. Each AS front end for a given security domain must use a different AS identifier.
`Sequence`	**Ten bits.** This field is split between the two bytes, passed with the most significant bits first. It represents the sequence of the OID assignment by the AS in the specified time unit indicated by
`Timestamp`	**Four bytes.** The timestamp of the generation of the OID in seconds, using the standard UNIX time (based January 1, 1970). The timestamp will wrap, but only after many years. The wrapping does not cause a problem. After 68 years, a duplicate assignment is possible.

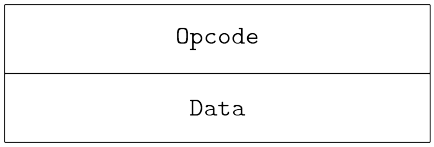
6.2. DOF Protocol Specification, OpenDOF TSC, [dof-2008-1] (7.0.1, 12 January 2018)

⇄ **General Application Header `dof-2008-1-pdu-13`**



`Application ID`	**Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.Contains the application identifier for the enclosed data.
`Application Data`	**Optional, variable length.** Contents are defined by the application.

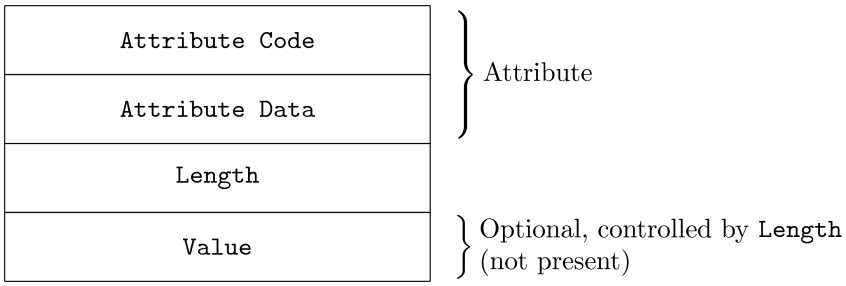
DSP Command/Response Format `dof-2008-1-pdu-15`



`Opcode` **One byte.** The value indicates the particular operation.

`Data` **Optional, variable length.** Each `Opcode` will define the contents of this field.

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.

Configuration Negative Acknowledge `dof-2008-1-pdu-19`

RESPONSE
Instance of DSP Command/Response Format`dof-2008-1-pdu-15`



`Opcode` **One byte.**

`Options` **Array of** Attribute/Value Pair`dof-2008-1-pdu-16`.This field list contains the requested options.

⇄ **Configuration Reject `dof-2008-1-pdu-20`**

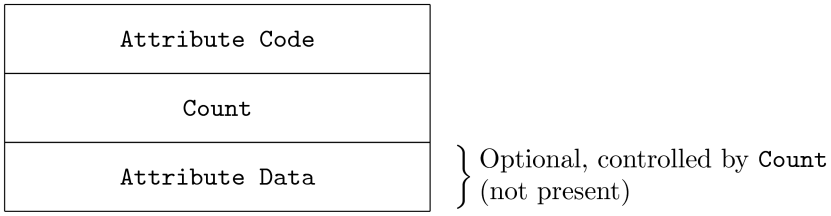
RESPONSE
Instance of DSP Command/Response Format`dof-2008-1-pdu-15`



`Opcode` **One byte.**

`Attributes` **Array of** Attribute Data List`dof-2008-1-pdu-21`.This field list contains the rejected attributes.

⇄ **Attribute Data List `dof-2008-1-pdu-21`**



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

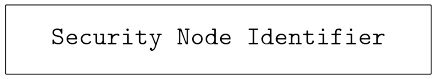
`Count` **Two bytes, MSB.** This field determines the length of the `Attribute Data`. It may be zero, in which case the `Attribute Data` field is not present.

`Attribute Data` **Two bytes, MSB.** This list contains Attribute Data values that are understood and are acceptable with the associated Attribute Code.

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

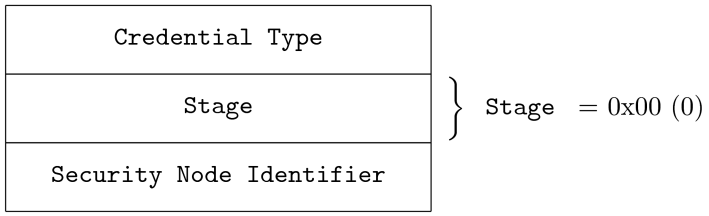
⇄ **Security Node Identifier `dof-2008-16-pdu-1`**

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`

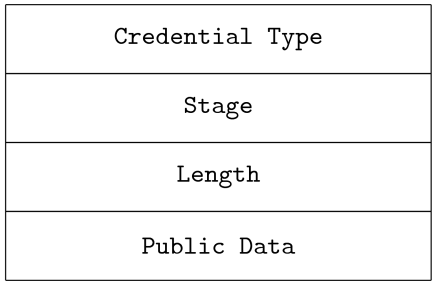


`Credential Type` **Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.

`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**.

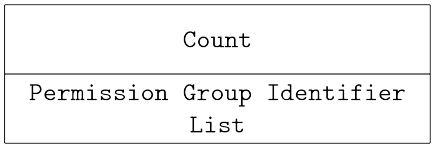
Security Domain `dof-2008-16-pdu-5`

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.

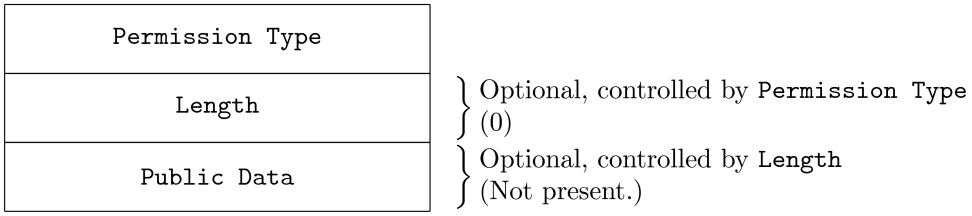
Security Scope `dof-2008-16-pdu-6`



`Count` **Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.Specifies the length of the **Permission Group Identifiers**).

`Permission Group Identifier List` **Array of** Compressed Unsigned 32-Bit`dof-2009-1-pdu-3`.**Optional, controlled by** `Count`.

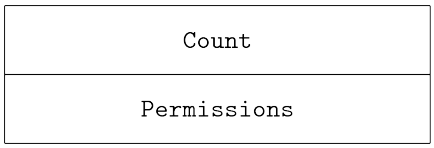
Permission `dof-2008-16-pdu-7`



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

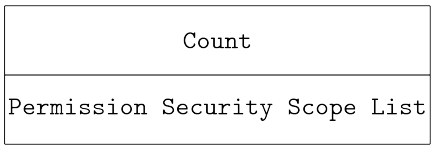
`Length`	**Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.**Optional, controlled by** `Permission Type`. Controls presence of **Data**. Present only if the **Permission Type** is odd. This field controls the size of **Data**.
`Public Data`	**Variable length. Optional, controlled by** `Length`. Present only if **Length** is greater than zero. The permission data depends on the **Permission Type**.

⇄ Permission Request `dof-2008-16-pdu-8`



`Count`	**Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.Controls length of **Permission List**.
`Permissions`	**Array of** Permission`dof-2008-16-pdu-7`.**Optional, controlled by** `Count`. Defines a set of permissions. The ordering of permissions inside the request does not affect the meaning and must not influence the result. However, the ordering must be stable because the security scope results arrive in the same order as the request.

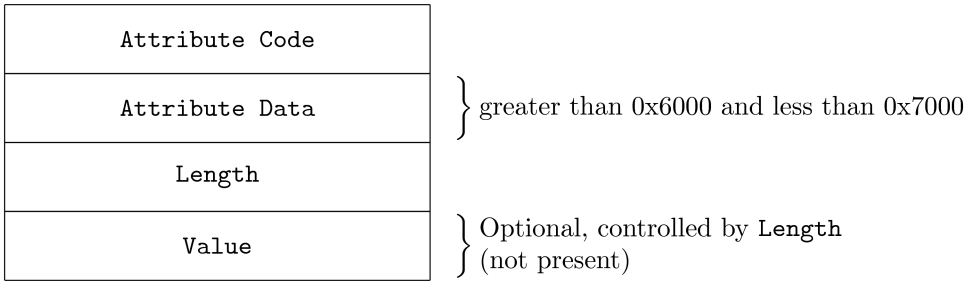
⇄ Permission Validation `dof-2008-16-pdu-9`



`Count`	**Instance of** Compressed Unsigned 16-Bit`dof-2009-1-pdu-1`.Controls length of **Scopes**.
`Permission Security Scope List`	**Array of** Security Scope`dof-2008-16-pdu-6`.**Optional, controlled by** `Count`. This is a parallel list to the corresponding permission request.

⇄ Security Mode of Operation Negotiation `dof-2008-16-pdu-11`

Instance of Attribute/Value Pair`dof-2008-1-pdu-16`



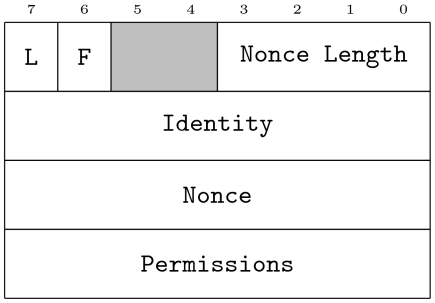
`Attribute Code` **One byte.**

`Attribute Data` **Two bytes, MSB.**

`Length` **One byte.**

`Value` **Variable length. Optional, controlled by** `Length`.

Key Request `dof-2008-16-pdu-13`



`L` **One bit.** Set to indicate a new authentication.

`F` **One bit.** Set if the request represents a fixed permission set. A node that requests a fixed permission set should not attempt to renegotiate its permissions. This means that the other node can optimize its own permission requests based on the current state of the other node's permissions.

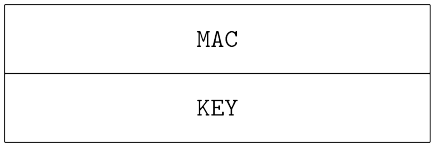
`Nonce Length` **Four bits.** Indicates the length of the `Nonce Length` field.

`Identity` **Instance of** Base Credential Format`dof-2008-16-pdu-2`.The identity of the node requesting authentication.

`Nonce` **Variable length of {{ln.name}}+1 bytes.** This number uniquely identifies this request for authentication.

`Permissions` **Instance of** Permission Request`dof-2008-16-pdu-8`.The permissions requested by the node.

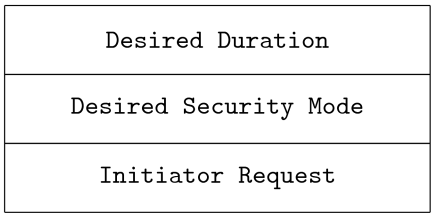
Key Grant `dof-2008-16-pdu-14`



`MAC` **Thirty-two bytes.** A hash described in the *DOF Security Specification*.

`KEY` **Thirty-two bytes.** The encrypted MAC field, convoluted with the new session key as described in the *DOF Security Specification*.

Session Initiator Block `dof-2008-16-pdu-15`

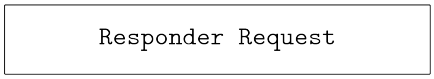


`Desired Duration` **One byte.** The desired duration of the secure session. The field is represented in minutes, with eight minute units. This represents the desired expiration time for the associated permission grants and session key.

`Desired Security Mode` **Instance of** Security Mode of Operation Negotiation`dof-2008-16-pdu-11`.The desired security mode of operation.

`Initiator Request` **Instance of** Key Request`dof-2008-16-pdu-13`.Initiator authentication information and permissions.

Session Responder Block `dof-2008-16-pdu-16`



`Responder Request` **Instance of** Key Request`dof-2008-16-pdu-13`.Responder authentication information and permissions.

Authentication Response Block `dof-2008-16-pdu-17`

Granted Duration
Session Security Scope
Initiator Validation
Responder Validation

`Granted Duration` **One byte.** The granted duration of the secure session. The field is represented in minutes, with eight minute units. The granted duration can be less than, equal to or greater than the requested duration. This represents the actual expiration time for the granted permissions and session key.

`Session Security Scope` **Instance of** Security Scope`dof-2008-16-pdu-6`.The security scope of the session.

`Initiator Validation` **Instance of** Permission Validation`dof-2008-16-pdu-9`.The validation block for the associated initiator request.

`Responder Validation` **Instance of** Permission Validation`dof-2008-16-pdu-9`.The validation block for the associated responder request.