Stream: Internet Engineering Task Force (IETF)
RFC: 9327
Category: Historic
Published: November 2022
ISSN: 2070-1721
Author: B. Haberman, Ed.
JHU

RFC 9327
Control Messages Protocol for Use with Network Time Protocol Version 4

Abstract

This document describes the structure of the control messages that were historically used with the Network Time Protocol (NTP) before the advent of more modern control and management approaches. These control messages have been used to monitor and control the NTP application running on any IP network attached computer. The information in this document was originally described in Appendix B of RFC 1305. The goal of this document is to provide an updated description of the control messages described in RFC 1305 in order to conform with the updated NTP specification documented in RFC 5905.

The publication of this document is not meant to encourage the development and deployment of these control messages. This document is only providing a current reference for these control messages given the current status of RFC 1305.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for the historical record.

This document defines a Historic Document for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9327.

Copyright Notice

Table of Contents

1. Introduction

[RFC1305] describes a set of control messages for use within the Network Time Protocol (NTP) when a comprehensive network management solution was not available. The definitions of these control messages were not promulgated to [RFC5905] when NTP version 4 was documented. These messages were intended for use only in systems where no other management facilities were available or appropriate, such as in dedicated-function bus peripherals. Support for these messages is not required in order to conform to [RFC5905]. The control messages are described here as a current reference for use with an implementation of NTP from RFC 5905.

The publication of this document is not meant to encourage the development and deployment of these control messages. This document is only providing a current reference for these control messages given the current status of RFC 1305.

1.1. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Control Message Overview

The NTP mode 6 control messages are used by NTP management programs (e.g., ntpq) when a more robust network management facility (e.g., SNMP) is not available. These control messages provide rudimentary control and monitoring functions to manage a running instance of an NTP server. These commands are not designed to be used for communication between instances of running NTP servers.

The NTP control message has the value 6 specified in the mode field of the first octet of the NTP header and is formatted as shown in Figure 1. The format of the data field is specific to each command or response; however, in most cases, the format is designed to be constructed and viewed by humans and so is coded in free-form ASCII. This facilitates the specification and

implementation of simple management tools in the absence of fully evolved network-management facilities. As in ordinary NTP messages, the authenticator field follows the data field. If the authenticator is used, the data field is zero-padded to a 32-bit boundary, but the padding bits are not considered part of the data field and are not included in the field count.

IP hosts are not required to reassemble datagrams over a certain size (576 octets for IPv4 [RFC0791] and 1280 octets for IPv6 [RFC8200]); however, some commands or responses may involve more data than will fit into a single datagram. Accordingly, a simple reassembly feature is included in which each octet of the message data is numbered starting with zero. As each fragment is transmitted, the number of its first octet is inserted in the offset field and the number of octets is inserted in the count field. The more-data (M) bit is set in all fragments except the last.

Most control functions involve sending a command and receiving a response, perhaps involving several fragments. The sender chooses a distinct, nonzero sequence number and sets the status field, "R" bit, and "E" bit to zero. The responder interprets the opcode and additional information in the data field, updates the status field, sets the "R" bit to one and returns the three 32-bit words of the header along with additional information in the data field. In the case of invalid message format or contents, the responder inserts a code in the status field, sets the "R" and "E" bits to one and, optionally, inserts a diagnostic message in the data field.

Some commands read or write system variables (e.g., s.offset) and peer variables (e.g., p.stratum) for an association identified in the command. Others read or write variables associated with a radio clock or other device directly connected to a source of primary synchronization information. To identify which type of variable and association, the Association ID is used. System variables are indicated by the identifier zero. As each association is mobilized a unique, nonzero identifier is created for it. These identifiers are used in a cyclic fashion, so that the chance of using an old identifier that matches a newly created association is remote. A management entity can request a list of current identifiers and subsequently use them to read and write variables for each association. An attempt to use an expired identifier results in an exception response, following which the list can be requested again.

Some exception events, such as when a peer becomes reachable or unreachable, occur spontaneously and are not necessarily associated with a command. An implementation may elect to save the event information for later retrieval, to send an asynchronous response (called a trap), or both. In case of a trap, the IP address and port number are determined by a previous command and the sequence field is set as described below. Current status and summary information for the latest exception event is returned in all normal responses. Bits in the status field indicate whether an exception has occurred since the last response and whether more than one exception has occurred.

Commands need not necessarily be sent by an NTP peer, so ordinary access-control procedures may not apply; however, the optional mask/match mechanism suggested in Section 6 provides the capability to control access by mode number, so this could be used to limit access for control messages (mode 6) to selected address ranges.

1.3. Remote Facility Message Overview

The original development of the NTP daemon included a Remote Facility for monitoring and configuration. This facility used mode 7 commands to communicate with the NTP daemon. This document illustrates the mode 7 packet format only. The commands embedded in the mode 7 messages are implementation specific and not standardized in any way. The mode 7 message format is described in Appendix A.

2. NTP Control Message Format

The format of the NTP Control Message header, which immediately follows the UDP header, is shown in Figure 1. Following the figure is a description of its header fields.

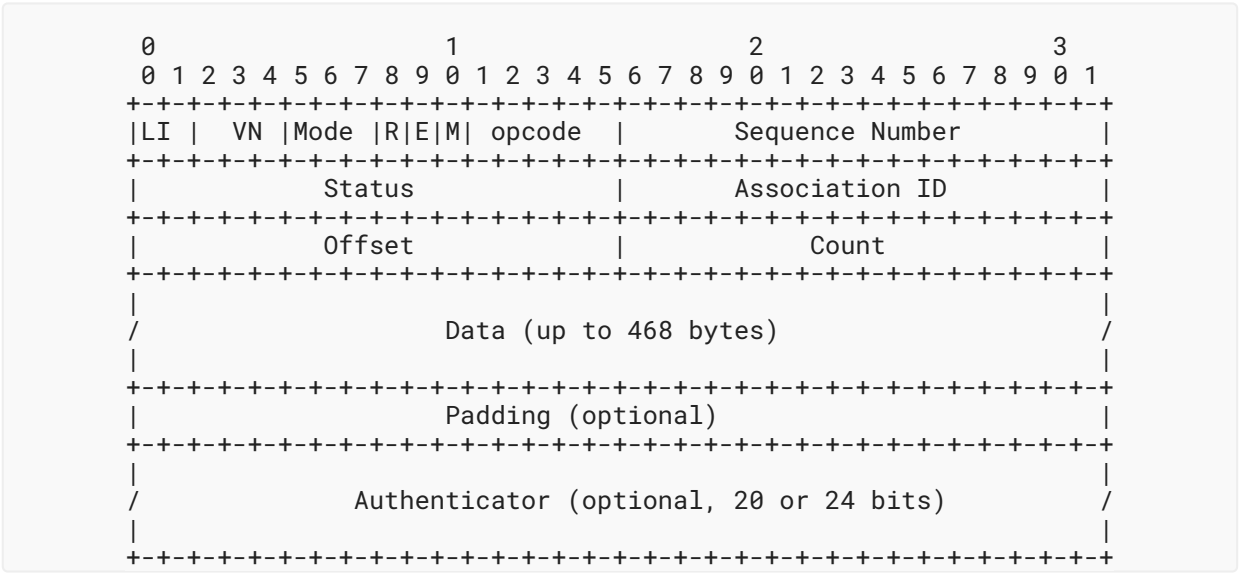

Figure 1: NTP Control Message Header

Leap Indicator (LI):
: This is a 2-bit integer that is set to b00 for control message requests and responses. The Leap Indicator value used at this position in most NTP modes is in the system status word provided in some control message responses.

Version Number (VN):
: This is a 3-bit integer indicating a minimum NTP version number. NTP servers do not respond to control messages with an unrecognized version number. Requests may intentionally use a lower version number to enable interoperability with earlier versions of NTP. Responses carry the same version as the corresponding request.

Mode:
: This is a 3-bit integer indicating the mode. The value 6 indicates an NTP control message.

Response Bit (R):
: Set to zero for commands; set to one for responses.

Error Bit (E):
: Set to zero for normal responses; set to one for an error response.

More Bit (M):
: Set to zero for the last fragment; set to one for all others.

Operation Code (opcode):
: This is a 5-bit integer specifying the command function. Values currently defined include the following:

Code	Meaning
0	reserved
1	read status command/response
2	read variables command/response
3	write variables command/response
4	read clock variables command/response
5	write clock variables command/response
6	set trap address/port command/response
7	trap response
8	runtime configuration command/response
9	export configuration to file command/response
10	retrieve remote address stats command/response
11	retrieve ordered list command/response
12	request client-specific nonce command/response
13-30	reserved
31	unset trap address/port command/response

Table 1: Operation Codes

Sequence Number:
: This is a 16-bit integer indicating the sequence number of the command or response. Each request uses a different sequence number. Each response carries the same sequence number as its corresponding request. For asynchronous trap responses, the responder increments the sequence number by one for each response, allowing trap receivers to detect missing trap responses. The sequence number of each fragment of a multiple-datagram response carries the same sequence number, copied from the request.

Status:
: This is a 16-bit code indicating the current status of the system, peer, or clock with values coded as described in following sections.

Association ID:
: This is a 16-bit unsigned integer identifying a valid association or zero for the system clock.

Offset:
: This is a 16-bit unsigned integer indicating the offset, in octets, of the first octet in the data area. The offset is set to zero in requests. Responses spanning multiple datagrams use a positive offset in all but the first datagram.

Count:
: This is a 16-bit unsigned integer indicating the length of the data field, in octets.

Data:
: This contains the message data for the command or response. The maximum number of data octets is 468.

Padding (optional):
: Contains zero to 3 octets with a value of zero, as needed to ensure the overall control message size is a multiple of 4 octets.

Authenticator (optional):
: When the NTP authentication mechanism is implemented, this contains the authenticator information defined in Appendix C of [RFC1305].

3. Status Words

Status words indicate the present status of the system, associations, and clock. They are designed to be interpreted by network-monitoring programs and are in one of four 16-bit formats shown in Figure 2 and described in this section. System and peer status words are associated with responses for all commands except the read clock variables, write clock variables, and set trap address/port commands. The association identifier zero specifies the system status word, while a nonzero identifier specifies a particular peer association. The status word returned in response to read clock variables and write clock variables commands indicates the state of the clock hardware and decoding software. A special error status word is used to report malformed command fields or invalid values.

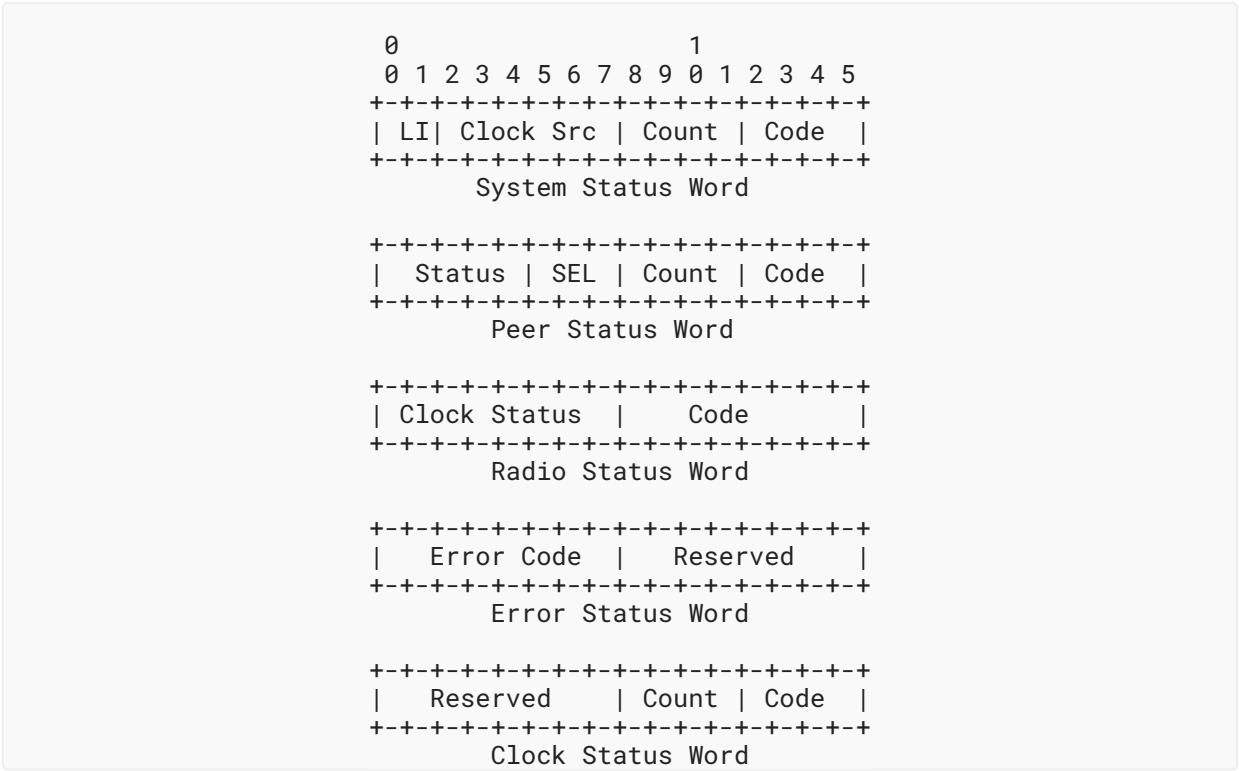

Figure 2: Status Word Formats

3.1. System Status Word

The system status word appears in the status field of the response to a read status or read variables command with a zero association identifier. The format of the system status word is as follows:

Leap Indicator (LI):
: This is a 2-bit code warning of an impending leap second to be inserted/deleted in the last minute of the current day, with bit 0 and bit 1, respectively, coded as follows:

LI	Meaning
00	no warning
01	insert second after 23:59:59 of the current day
10	delete second 23:59:59 of the current day
11	unsynchronized

Table 2: Leap Indicator Codes

Clock Source (Clock Src):
: This is a 6-bit integer indicating the current synchronization source, with values coded as follows:

Code	Meaning
0	unspecified or unknown
1	Calibrated atomic clock (e.g., PPS, HP 5061)
2	VLF (band 4) or LF (band 5) radio (e.g., OMEGA,, WWVB)
3	HF (band 7) radio (e.g., CHU, MSF, WWV/H)
4	UHF (band 9) satellite (e.g., GOES, GPS)
5	local net (e.g., DCN, TSP, DTS)
6	UDP/NTP
7	UDP/TIME
8	eyeball-and-wristwatch
9	telephone modem (e.g., NIST)
10-63	reserved

Table 3: Clock Source Values

System Event Counter (Count):
: This is a 4-bit integer indicating the number of system events occurring since the last time the System Event Code changed. Upon reaching 15, subsequent events with the same code are not counted.

System Event Code (Code):
: This is a 4-bit integer identifying the latest system exception event, with new values overwriting previous values, and coded as follows:

Code	Meaning
0	unspecified
1	frequency correction (drift) file not available
2	frequency correction started (frequency stepped)
3	spike detected and ignored, starting stepout timer

Code	Meaning
4	frequency training started
5	clock synchronized
6	system restart
7	panic stop (required step greater than panic threshold)
8	no system peer
9	leap second insertion/deletion armed for the current month
10	leap second disarmed
11	leap second inserted or deleted
12	clock stepped (stepout timer expired)
13	kernel loop discipline status changed
14	leapseconds table loaded from file
15	leapseconds table outdated, updated file needed

Table 4: System Event Codes

3.2. Peer Status Word

A peer status word is returned in the status field of a response to a read status, read variables, or write variables command and appears in the list of Association IDs and status words returned by a read status command with a zero Association ID. The format of a peer status word is as follows:

Peer Status (Status):
: This is a 5-bit code indicating the status of the peer determined by the packet procedure, with bits assigned as follows:

Peer Status bit	Meaning
0	configured (peer.config)
1	authentication enabled (peer.authenable)
2	authentication okay (peer.authentic)
3	reachability okay (peer.reach != 0)

Peer Status bit	Meaning
4	broadcast association

Table 5: Peer Status Bits

Peer Selection (SEL):
: This is a 3-bit integer indicating the status of the peer determined by the clock-selection procedure, with values coded as follows:

Sel	Meaning
0	rejected
1	discarded by intersection algorithm
2	discarded by table overflow (not currently used)
3	discarded by the cluster algorithm
4	included by the combine algorithm
5	backup source (with more than sys.maxclock survivors)
6	system peer (synchronization source)
7	PPS (pulse per second) peer

Table 6: Peer Selection Values

Peer Event Counter (Count):
: This is a 4-bit integer indicating the number of peer exception events that occurred since the last time the peer event code changed. Upon reaching 15, subsequent events with the same code are not counted.

Peer Event Code (Code):
: This is a 4-bit integer identifying the latest peer exception event, with new values overwriting previous values, and coded as follows:

Peer Event Code	Meaning
0	unspecified
1	association mobilized
2	association demobilized
3	peer unreachable (peer.reach was nonzero now zero)

Peer Event Code	Meaning
4	peer reachable (peer.reach was zero now nonzero)
5	association restarted or timed out
6	no reply (only used with one-shot clock set command)
7	peer rate limit exceeded (kiss code RATE received)
8	access denied (kiss code DENY received)
9	leap second insertion/deletion at month's end armed by peer vote
10	became system peer (sys.peer)
11	reference clock event (see clock status word)
12	authentication failed
13	popcorn spike suppressed by peer clock filter register
14	entering interleaved mode
15	recovered from interleave error

Table 7: Peer Event Code Values

3.3. Clock Status Word

There are two ways a reference clock can be attached to an NTP service host: as a dedicated device managed by the operating system and as a synthetic peer managed by NTP. As in the read status command, the Association ID is used to identify the correct variable for each clock: zero for the system clock and nonzero for a peer clock. Only one system clock is supported by the protocol, although many peer clocks can be supported. A system or peer clock status word appears in the status field of the response to a read clock variables or write clock variables command. This word can be considered to be an extension of the system status word or the peer status word as appropriate. The format of the clock status word is as follows:

Reserved:
: This is an 8-bit integer that is ignored by requesters and zeroed by responders.

Count:
: This is a 4-bit integer indicating the number of clock events that occurred since the last time the clock event code changed. Upon reaching 15, subsequent events with the same code are not counted.

Clock Code (Code):
: This is a 4-bit integer indicating the current clock status, with values coded as follows:

Clock Status	Meaning
0	clock operating within nominals
1	reply timeout
2	bad reply format
3	hardware or software fault
4	propagation failure
5	bad date format or value
6	bad time format or value
7-15	reserved

Table 8: Clock Code Values

3.4. Error Status Word

An error status word is returned in the status field of an error response as the result of invalid message format or contents. Its presence is indicated when the E (error) bit is set along with the response (R) bit in the response. It consists of an 8-bit integer coded as follows:

Error Status	Meaning
0	unspecified
1	authentication failure
2	invalid message length or format
3	invalid opcode
4	unknown Association ID
5	unknown variable name
6	invalid variable value
7	administratively prohibited
8-255	reserved

Table 9: Error Status Word Codes

4. Commands

Commands consist of the header and optional data field shown in Figure 1. When present, the data field contains a list of identifiers or assignments in the form <<identifier>>[=<<value>>],<<identifier>>[=<<value>>],... where <<identifier>> is the ASCII name of a system or peer variable such as the ones specified in RFC 5905 and <<value>> is expressed as a decimal, hexadecimal, or string constant in the syntax of the C programming language. Where no ambiguity exists, the "sys." or "peer." prefixes can be suppressed. Space characters (ASCII nonprinting format effectors) can be added to improve readability for simple monitoring programs that do not reformat the data field. Representations of note are as follows:

- IPv4 internet addresses are written in the form [n.n.n.n], where n is in decimal notation and the brackets are optional
- IPv6 internet addresses are formulated based on the guidelines defined in [RFC5952].
- Timestamps (including reference, originate, receive, and transmit values) and the logical clock are represented in units of seconds and fractions, preferably in hexadecimal notation.
- Delay, offset, dispersion, and distance values are represented in units of milliseconds and fractions, preferably in decimal notation.
- All other values are represented as is, preferably in decimal notation.

Implementations may define variables other than those described in RFC 5905; called "extramural variables", these are distinguished by the inclusion of some character type other than alphanumeric or "." in the name. For those commands that return a list of assignments in the response data field, if the command data field is empty, it is expected that all available variables defined in RFC 5905 will be included in the response. For the read commands, if the command data field is nonempty, an implementation may choose to process this field to individually select which variables are to be returned.

Commands are interpreted as follows:

Read Status (1):
: The command data field is empty or contains a list of identifiers separated by commas. The command operates in two ways depending on the value of the Association ID. If this identifier is nonzero, the response includes the peer identifier and status word. Optionally, the response data field may contain other information, such as described in the Read Variables command. If the association identifier is zero, the response includes the system identifier (0) and status word; the data field contains a list of binary-coded pairs <<Association ID>> <<status word>>, one for each currently defined association.

Read Variables (2):
: The command data field is empty or contains a list of identifiers separated by commas. If the Association ID is nonzero, the response includes the requested peer identifier and status word; the data field contains a list of peer variables and values as described above. If the

Association ID is zero, the data field contains a list of system variables. If a peer has been selected as the synchronization source, the response includes the peer identifier and status word; otherwise, the response includes the system identifier (0) and status word.

Write Variables (3):
The command data field contains a list of assignments as described above. The variables are updated as indicated. The response is as described for the Read Variables command.

Read Clock Variables (4):
The command data field is empty or contains a list of identifiers separated by commas. The Association ID selects the system clock variables or peer clock variables in the same way as in the Read Variables command. The response includes the requested clock identifier and status word; the data field contains a list of clock variables and values, including the last timecode message received from the clock.

Write Clock Variables (5):
The command data field contains a list of assignments as described above. The clock variables are updated as indicated. The response is as described for the read clock variables command.

Set Trap Address/Port (6):
The command Association ID, status, and data fields are ignored. The address and port number for subsequent trap messages are taken from the source address and port of the control message itself. The initial trap counter for trap response messages is taken from the sequence field of the command. The response association identifier, status, and data fields are not significant. Implementations should include logical timeouts that prevent trap transmissions if the monitoring program does not renew this information after a lengthy interval.

Trap Response (7):
This message is sent when a system, peer, or clock exception event occurs. The opcode field is 7 and the R bit is set. The trap counter is incremented by one for each trap sent and the sequence field set to that value. The trap message is sent using the IP address and port fields established by the set trap address/port command. If a system trap, the Association ID field is set to zero and the status field contains the system status word. If a peer trap, the Association ID field is set to that peer and the status field contains the peer status word. Optional ASCII-coded information can be included in the data field.

Configure (8):
The command data is parsed and applied as if supplied in the daemon configuration file.

Save Configuration (9):
Writes a snapshot of the current configuration to the file name supplied as the command data. Further, the command is refused unless a directory in which to store the resulting files has been explicitly configured by the operator.

Read Most Recently Used (MRU) list (10):
Retrieves records of recently seen remote addresses and associated statistics. This command supports all of the state variables defined in Section 9 of [RFC5905]. Command data consists of name=value pairs controlling the selection of records, as well as a requestor-specific nonce

previously retrieved using this command or opcode 12 (Request Nonce). The response consists of name=value pairs where some names can appear multiple times using a dot followed by a zero-based index to distinguish them and to associate elements of the same record with the same index. A new nonce is provided with each successful response.

Read ordered list (11):
: Retrieves a list ordered by IP address (IPv4 information precedes IPv6 information). If the command data is empty or is the seven characters "ifstats", the associated statistics, status, and counters for each local address are returned. If the command data is the characters "addr_restrictions", then the set of IPv4 remote address restrictions followed by the set of IPv6 remote address restrictions (access control lists) are returned. Other command data returns error code 5 (unknown variable name). Similar to Read MRU, response information uses zero-based indexes as part of the variable name preceding the equals sign and value, where each index relates information for a single address or network. This opcode requires authentication.

Request Nonce (12):
: Retrieves a 96-bit nonce specific to the requesting remote address, which is valid for a limited period. Command data is not used in the request. The nonce consists of a 64-bit NTP timestamp and 32 bits of hash derived from that timestamp, the remote address, and salt known only to the server, which varies between daemon runs. Inclusion of the nonce by a management agent demonstrates to the server that the agent can receive datagrams sent to the source address of the request, making source address "spoofing" more difficult in a similar way as TCP's three-way handshake.

Unset Trap (31):
: Removes the requesting remote address and port from the list of trap receivers. Command data is not used in the request. If the address and port are not in the list of trap receivers, the error code is 4 (bad association).

5. IANA Considerations

This document has no IANA actions.

6. Security Considerations

A number of security vulnerabilities have been identified with these control messages.

NTP's control query interface allows reading and writing of system, peer, and clock variables remotely from arbitrary IP addresses using commands mentioned in Section 4. Overwriting these variables, but not reading them, requires authentication by default. However, this document argues that an NTP host must authenticate all control queries and not just ones that overwrite these variables. Alternatively, the host can use an access control list to explicitly list IP addresses that are allowed to control query the clients. These access controls are required for the following reasons:

NTP as a Distributed Denial-of-Service (DDoS) vector:
: NTP timing query and response packets (modes 1-2, 3-4, and 5) are usually short in size. However, some NTP control queries generate a very long packet in response to a short query. As such, there is a history of use of NTP's control queries, which exhibit such behavior, to perform DoS attacks. These off-path attacks exploit the large size of NTP control queries to cause UDP-based amplification attacks (e.g., mode 7 monlist command generates a very long packet in response to a small query [CVE-DOS]). These attacks only use NTP as a vector for DoS attacks on other protocols, but do not affect the time service on the NTP host itself. To limit the sources of these malicious commands, NTP server operators are recommended to deploy ingress filtering [RFC3704].

Time-shifting attacks through information leakage/overwriting:
: NTP hosts save important system and peer state variables. An off-path attacker who can read these variables remotely can leverage the information leaked by these control queries to perform time-shifting and DDoS attacks on NTP clients. These attacks do affect time synchronization on the NTP hosts. For instance:

- In the client/server mode, the client stores its local time when it sends the query to the server in its xmt peer variable. This variable is used to perform TEST2 to non-cryptographically authenticate the server (i.e., if the origin timestamp field in the corresponding server response packet matches the xmt peer variable, then the client accepts the packet). An off-path attacker with the ability to read this variable can easily spoof server response packets for the client, which will pass TEST2 and can deny service or shift time on the NTP client. The specific attack is described in [CVE-SPOOF].
- The client also stores its local time when the server response is received in its rec peer variable. This variable is used for authentication in interleaved-pivot mode. An off-path attacker with the ability to read this state variable can easily shift time on the client by passing this test. This attack is described in [CVE-SHIFT].

Fast-Scanning:
: NTP mode 6 control messages are usually small UDP packets. Fast-scanning tools like ZMap can be used to spray the entire (potentially reachable) Internet with these messages within hours to identify vulnerable hosts. To make things worse, these attacks can be extremely low-rate, only requiring a control query for reconnaissance and a spoofed response to shift time on vulnerable clients.

The mode 6 and 7 messages are vulnerable to replay attacks [CVE-Replay]:
: If an attacker observes mode 6/7 packets that modify the configuration of the server in any way, the attacker can apply the same change at any time later by simply sending the packets to the server again. The use of the nonce (Request Nonce command) provides limited protection against replay attacks.

NTP best practices recommend configuring NTP with the no-query parameter. The no-query parameter blocks access to all remote control queries. However, sometimes the hosts do not want to block all queries and want to give access for certain control queries remotely. This could be for the purpose of remote management and configuration of the hosts in certain scenarios. Such hosts tend to use firewalls or other middleboxes to blacklist certain queries within the network.

Significantly fewer hosts respond to mode 7 monlist queries as compared to other control queries because it is a well-known and exploited control query. These queries are likely blocked using blacklists on firewalls and middleboxes rather than the no-query option on NTP hosts. The remaining control queries that can be exploited likely remain out of the blacklist because they are undocumented in the current NTP specification [RFC5905].

This document describes all of the mode 6 control queries allowed by NTP and can help administrators make informed decisions on security measures to protect NTP devices from harmful queries and likely make those systems less vulnerable. The use of the legacy mode 6 interface is **NOT RECOMMENDED**. Regardless of which mode 6 commands an administrator may elect to allow, remote access to this facility needs to be protected from unauthorized access (e.g., strict Access Control Lists (ACLs)). Additionally, the legacy interface for mode 6 commands **SHOULD NOT** be utilized in new deployments or implementation of NTP.

7. References

7.1. Normative References

[RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, DOI 10.17487/RFC1305, March 1992, <https://www.rfc-editor.org/info/rfc1305>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <https://www.rfc-editor.org/info/rfc3704>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[CVE-DOS] NIST National Vulnerability Database, "CVE-2013-5211 Detail", 2 January 2014, <https://nvd.nist.gov/vuln/detail/CVE-2013-5211>.

[CVE-Replay] NIST National Vulnerability Database, "CVE-2015-8140 Detail", 30 January 2015, <https://nvd.nist.gov/vuln/detail/CVE-2015-8140>.

[CVE-SHIFT] NIST National Vulnerability Database, "CVE-2016-1548 Detail", 6 January 2017, <https://nvd.nist.gov/vuln/detail/CVE-2016-1548>.

[CVE-SPOOF] NIST National Vulnerability Database, "CVE-2015-8139 Detail", 30 January 2017, <https://nvd.nist.gov/vuln/detail/CVE-2015-8139>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

Appendix A. NTP Remote Facility Message Format

The format of the NTP Remote Facility Message header, which immediately follows the UDP header, is shown in Figure 3. A description of its fields follows Figure 3. Bit positions marked as zero are reserved and should always be transmitted as zero.

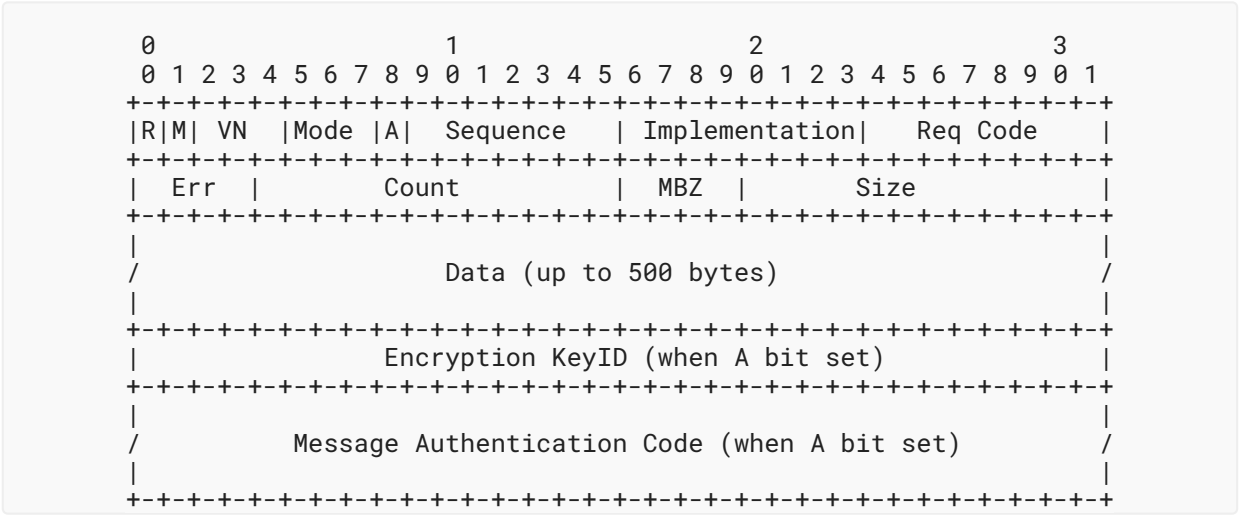

Figure 3: NTP Remote Facility Message Header

Response Bit (R):
: Set to 0 if the packet is a request. Set to 1 if the packet is a response.

More Bit (M):
: Set to 0 if this is the last packet in a response; otherwise, set to 1 in responses requiring more than one packet.

Version Number (VN):
: Set to the version number of the NTP daemon.

Mode:
: Set to 7 for Remote Facility messages.

Authenticated Bit (A):
: If set to 1, this packet contains authentication information.

Sequence:
: For a multi-packet response, this field contains the sequence number of this packet. Packets in a multi-packet response are numbered starting with 0. The More Bit is set to 1 for all packets but the last.

Implementation:
: The version number of the implementation that defined the request code used in this message. An implementation number of 0 is used for a request code supported by all versions of the NTP daemon. The value 255 is reserved for future extensions.

Request Code (Req Code):
: An implementation-specific code that specifies the operation being requested. A request code definition includes the format and semantics of the data included in the packet.

Error (Err):
: Set to 0 for a request. For a response, this field contains an error code relating to the request. If the Error is nonzero, the operation requested wasn't performed.

  0: no error

  1: incompatible implementation number

  2: unimplemented request code

  3: format error

  4: no data available

  7: authentication failure

Count:
: The number of data items in the packet. Range is 0 to 500.

Must Be Zero (MBZ):
: A reserved field set to 0 in requests and responses.

Size:
: The size of each data item in the packet. Range is 0 to 500.

Data:
: A variable-sized field containing request/response data. For requests and responses, the size in octets must be greater than or equal to the product of the number of data items (Count) and the size of a data item (Size). For requests, the data area is exactly 40 octets in length. For responses, the data area will range from 0 to 500 octets, inclusive.

Encryption KeyID:
: A 32-bit unsigned integer used to designate the key used for the Message Authentication Code. This field is included only when the A bit is set to 1.

Message Authentication Code:
: An optional Message Authentication Code defined by the version of the NTP daemon indicated in the Implementation field. This field is included only when the A bit is set to 1.

Acknowledgements

Tim Plunkett created the original version of this document. Aanchal Malhotra provided the initial version of the Security Considerations section.

Karen O'Donoghue, David Hart, Harlan Stenn, and Philip Chimento deserve credit for portions of this document due to their earlier efforts to document these commands.

Miroshav Lichvar, Ulrich Windl, Dieter Sibold, J Ignacio Alvarez-Hamelin, and Alex Campbell provided valuable comments on various draft versions of this document.

Contributors

Dr. David Mills specified the vast majority of the mode 6 commands during the development of [RFC1305] and deserves the credit for their existence and use.

Author's Address

Brian Haberman (EDITOR)
JHU
Email: brian@innovationslab.net