Troubleshoot and Debug Network Time Protocol (NTP) Issues

Contents

Introduction

This document describes how to troubleshoot Network Time Protocol (NTP) issues with `debug` commands and the `show ntp` command.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

NTP show Commands

Before you look at the cause of NTP problems, you must understand the use of and output from these

commands:

- show ntp association
- show ntp association detail
- show ntp status

Note: Use the Command Lookup Tool in order to obtain more information on the commands used in this section. Only registered Cisco users can access internal tools and information.

Note: The Output Interpreter Tool supports certain show commands. Use the Output Interpreter Tool in order to view an analysis of show command output. Only registered Cisco users can access internal tools and information.

show ntp association

An NTP association can be:

- a peer association (one system is willing to synchronize to the other system or to allow the other system to synchronize to it)
- a server association (only one system synchronizes to the other system and not the other way around).

This is an example of output from the show ntp association command:

```
CLA_PASA#sh ntp association
      address         ref clock     st  when  poll reach  delay  offset    disp
 ~10.127.7.1       10.127.7.1       9    50    64  377     0.0    0.00     0.0
 ~10.50.44.69      10.50.36.106      5  21231  1024   0      3.8   -4.26  16000.
+~10.50.44.101     10.50.38.114      5    57    64    1      3.6   -4.30  15875.
+~10.50.44.37      10.50.36.50       5     1   256  377     0.8    1.24     0.2
 ~10.50.44.133     10.50.38.170      5  12142  1024   0      3.2    1.24  16000.
+~10.50.44.165     10.50.38.178      5    35   256  357     2.5   -4.09     0.2
+~10.50.38.42      10.79.127.250     4     7   256  377     0.8   -0.29     0.2
*~10.50.36.42      10.79.127.250     4   188   256  377     0.7   -0.17     0.3
+~10.50.38.50      10.79.127.250     4    42   256  377     0.9    1.02     0.4
+~10.50.36.50      10.79.127.250     4    20   256  377     0.7    0.87     0.5
 * primary (synced), # primary (unsynced), + selected, - candidate, ~ configured
```

Term	Explanation
	Characters before the address have these definitions: * Synchronized to this peer # Almost synchronized to this peer + Peer selected for possible synchronization - Peer is a candidate for selection ~ Peer is statically configured
address	This is the IP address of the peer. In the example, the first entry shows 127.127.7.1. This indicates that the local machine has synced with itself. Generally, only an NTP primary syncs with itself.
ref	This is the address of the reference clock for the peer. In the example, the first six peers/servers

clock	have a private IP as the reference clock, so their primarys are probably routers, switches, or servers within the local network. For the last four entries, the reference clock is a public IP, so their primarys are probably a public time source.
st	NTP uses the concept of a stratum in order to describe how far away (in NTP hops) a machine is from an authoritative time source. For example, a stratum 1 time server has a radio or atomic clock directly attached to it. It sends its time to a stratum 2 time server through NTP, and so on up to stratum 16. A machine that runs NTP automatically chooses the machine with the lowest stratum number with which it can communicate and uses NTP as its time source.
when	The time since the last NTP packet was received from a peer is reported in seconds. This value must be lower than the polling interval.
poll	The polling interval is reported in seconds. The interval usually starts with a minimum of 64-second poll intervals. The RFC specifies that no more than one NTP transaction per minute is needed in order to synchronize two machines. As NTP becomes stable between a client and a server, the poll interval can increase in small steps from 64 seconds up to 1024 seconds and generally stabilizes somewhere in between. But, this value dynamically changes, based on the network conditions between the client and the server and the loss of NTP packets. If a server is unreachable for some time, the poll interval is increased in steps to 1024 seconds in order to reduce network overhead. It is not possible to adjust the NTP poll interval on a router, because the internal is determined by heuristic algorithms.
reach	Peer reachability is a bit string reported as an octal value. This field shows whether the last eight packets were received by the NTP process on the Cisco IOS® software. The packets must be received, processed, and accepted as valid by the NTP process and not just by the router or switch that receives the NTP IP packets. Reach uses the poll interval for a time out in order to decide whether a packet was received or not. The poll interval is the time that NTP waits before it concludes that a packet was lost. The poll time can be different for different peers, so the time before reach decides that a packet was lost can also different for different peers. In the example, there are four different reach values: • 377 octal = 11111111 binary, which indicates the NTP process received the last eight packets. • 0 octal = 00000000, which indicates the NTP process did not receive any packet. • 1 octal = 00000001, which indicates the NTP process received only the latest packet. • 357 octal = 11101111, which indicates the packet before the latest four packets was lost. Reach is a good indicator of whether NTP packets are dropped because of a poor link, CPU issues and other intermittent problems. Unit Converter is an online unit converter for this and many other conversions.
delay	The round-trip delay to peer is reported in milliseconds. In order to set the clock more accurately, this delay is taken into account when the clock time is set.

offset	Offset is the clock time difference between the peers or between the primary and client. This value is the correction that is applied to a client clock in order to synchronize it. A positive value indicates the server clock is higher. A negative value indicates the client clock is higher.

disp	Dispersion, reported in seconds, is the maximum clock time difference that was ever observed between the local clock and server clock. In the example, dispersion is 0.3 for the server 10.50.36.42, so the maximum time difference ever observed locally between the local clock and the server clock is 0.3 seconds.
	
	You can expect to see a high value when the clocks are sync initially. But, if the dispersion is too high at other times, the NTP process on the client does not accept NTP messages from the server. Maximum dispersion is 16000; in the example, that is the dispersion for servers 10.50.44.69 and 10.50.44.133, so the local client does not accept time from these servers.
	
	If the reach is zero and dispersion is very high, the client is probably does not accept messages from that server. Refer to the second line of the example:
	
	`     address      ref clock  st   when  poll reach  delay  offset   disp`
	`~10.50.44.69  10.50.36.106   5  21231  1024     0    3.8   -4.26  16000.`
	
	Even though the offset is just -4.26, the dispersion is very high (perhaps due to a past event), and the reach is zero, so this client does not accept time from this server.

show ntp association detail

This is an example of output from the show ntp association detail command:

```
Router#sho ntp assoc detail
10.4.2.254 configured, our_primary, sane, valid, stratum 1
ref ID .GPS., time D36968AA.CC528FE7 (02:10:50.798 UTC Fri May 25 2012)
our mode client, peer mode server, our poll intvl 64, peer poll intvl 64
root delay 0.00 msec, root disp 0.44, reach 377, sync dist 207.565
delay 2.99 msec, offset 268.3044 msec, dispersion 205.54
precision 2**19, version 3
org time D36968B7.E74172BF (02:11:03.903 UTC Fri May 25 2012)
rcv time D36968B7.A2F44E2C (02:11:03.636 UTC Fri May 25 2012)
xmt time D36968B7.A21D3780 (02:11:03.633 UTC Fri May 25 2012)
filtdelay =     2.99    2.88  976.61  574.65  984.71  220.26  168.12    2.72
filtoffset =  268.30  172.15 -452.49 -253.59 -462.03  -81.98  -58.04   22.38
filterror =     0.02    0.99    1.95    1.97    2.00    2.01    2.03    2.04

10.3.2.254 configured, selected, sane, valid, stratum 1
ref ID .GPS., time D36968BB.B16C4A21 (02:11:07.693 UTC Fri May 25 2012)
our mode client, peer mode server, our poll intvl 64, peer poll intvl 64
root delay 0.00 msec, root disp 3.34, reach 377, sync dist 192.169
delay 0.84 msec, offset 280.3251 msec, dispersion 188.42
precision 2**19, version 3
org time D36968BD.E69085E4 (02:11:09.900 UTC Fri May 25 2012)
rcv time D36968BD.9EE9048B (02:11:09.620 UTC Fri May 25 2012)
xmt time D36968BD.9EA943EF (02:11:09.619 UTC Fri May 25 2012)
filtdelay =     0.84    0.75  663.68    0.67    0.72  968.05  714.07    1.14
```

```
filtoffset =  280.33  178.13 -286.52   42.88   41.41 -444.37 -320.25   35.15
filterror =     0.02    0.99    1.97    1.98    1.98    2.00    2.03    2.03

10.1.2.254 configured, insane, invalid, stratum 1
ref ID .GPS., time D3696D3D.BBB4FF24 (02:30:21.733 UTC Fri May 25 2012)
our mode client, peer mode server, our poll intvl 64, peer poll intvl 64
root delay 0.00 msec, root disp 4.15, reach 1, sync dist 15879.654
delay 0.98 msec, offset 11.9876 msec, dispersion 15875.02
precision 2**19, version 3
org time D3696D3D.E4C253FE (02:30:21.893 UTC Fri May 25 2012)
rcv time D3696D3D.E1D0C1B9 (02:30:21.882 UTC Fri May 25 2012)
xmt time D3696D3D.E18A748D (02:30:21.881 UTC Fri May 25 2012)
filtdelay =     0.98    0.00    0.00    0.00    0.00    0.00    0.00    0.00
filtoffset =   11.99    0.00    0.00    0.00    0.00    0.00    0.00    0.00
filterror =     0.02 16000.0 16000.0 16000.0 16000.0 16000.0 16000.0 16000.0
```

Terms already defined in the show up association section are not repeated here.

<table>
<tr><th>Term</th><th>Explanation</th></tr>
<tr><td>configured</td><td>This NTP clock source has been configured to be a server. This value can also be dynamic, where the peer/server was dynamically discovered.</td></tr>
<tr><td>our_primary</td><td>The local client is synchronized to this peer.</td></tr>
<tr><td>selected</td><td>The peer/server is selected for possible synchronization, when 'our_primary' fails or the client loses sync.</td></tr>
<tr><td>sane</td><td>Sanity tests are used in order to test the NTP packet received from a server. These tests are specified in RFC 1305, Network Time Protocol (Version 3) Specification, Implementation and Analysis. The tests are:
<table>
<tr><th>Test</th><th>Mask</th><th>Explanation</th></tr>
<tr><td>1</td><td>0x01</td><td>Duplicate packet received</td></tr>
<tr><td>2</td><td>0x02</td><td>Bogus packet received</td></tr>
<tr><td>3</td><td>0x04</td><td>Protocol unsynchronized</td></tr>
<tr><td>4</td><td>0x08</td><td>Peer delay/dispersion failed boundary check</td></tr>
<tr><td>5</td><td>0x10</td><td>Peer authentication failed</td></tr>
<tr><td>6</td><td>0x20</td><td>Peer clock unsynchronized (common for unsynched server)</td></tr>
<tr><td>7</td><td>0x40</td><td>Peer stratum out of bound</td></tr>
<tr><td>8</td><td>0x80</td><td>Root delay/dispersion failed boundary check</td></tr>
</table>
Packet data is valid if tests 1 to 4 are passed. The data is then used in order to calculate offset, delay, and dispersion.

Packet header is valid if tests 5 to 8 are passed. Only packets with a valid header can be used to determine whether a peer can be selected for synchronization.</td></tr>
</table>

insane	The sanity checks have failed, so time from the server is not accepted. The server is unsynced.
valid	The peer/server time is valid. The local client accepts this time if this peer becomes the primary.
invalid	The peer/server time is invalid, and time cannot be accepted.
ref ID	Each peer/server is assigned a reference ID (label).
time	Time is the last time stamp received from that peer/server.
our mode/ peer mode	This is the state of the local client/peer.
our poll intvl/ peer poll intvl	This is the poll interval from our poll to this peer or from the peer to the local machine.
root delay	Root delay is the delay in milliseconds to the root of the NTP setup. Stratum 1 clocks are considered to be at the root of an NTP setup/design. In the example, all three servers can be the root because they are at stratum 1.
root dispersion	Root dispersion is the maximum clock time difference that was ever observed between the local clock and the root clock. Refer to the explanation of 'disp' under show up association for more details.
sync dist.	This is an estimate of the maximum difference between the time on the stratum 0 source and the time measured by the client; it consists of components for round trip time, system precision, and clock drift since the last actual read of the stratum source. In a large NTP setup (NTP servers at stratum 1 in the internet, with servers that source time at different strata) with servers/clients at multiple strata, NTP synchronization topology must be organized in order to produce the highest accuracy, but must never be allowed to form a time sync loop. An additional factor is that each increment in stratum involves a potentially unreliable time server, which introduces additional measurement errors. The selection algorithm used in NTP uses a variant of the Bellman-Ford distributed routing algorithm in order to compute the minimum-weight spanning trees rooted on the primary servers. The distance metric used by the algorithm consists of the stratum plus the synchronization distance, which itself consists of the dispersion plus one-half the absolute delay. Thus, the synchronization path always takes the minimum number of servers to the root; ties are resolved on the basis of maximum error.
delay	This is the round trip delay to peer.

precision	This is the precision of the peer clock in Hz.
version	This is the NTP version number used by the peer.
org time	This is the time stamp of the NTP packet originator; in other words, it is the peer time stamp when it created the NTP packet but before it sent the packet to the local client.

rcv time	This is the time stamp when the local client received the message. The difference between org time and rcv time is the offset for this peer. In the example, primary 10.4.2.254 has these times:
	
	`org time D36968B7.E74172BF (02:11:03.903 UTC Fri May 25 2012)`
	`rcv time D36968B7.A2F44E2C (02:11:03.636 UTC Fri May 25 2012)`
	
	The difference is the offset of 268.3044 msec.

xmt time	This is the transmit time stamp for the NTP packet the local client sends to this peer/server.

filtdelay
filtoffset
filterror
	This is the round trip delay in milliseconds of each sample.
	This is the clock offset in milliseconds of each sample.
	This is the approximate error of each sample.
	
	A sample is the last NTP packet received. In the example, primary 10.4.2.254 has these values:
	
	`filtdelay =  2.99  2.88  976.61  574.65  984.71 220.26 168.12  2.72`
	`filtoffset = 268.30 172.15 -452.49 -253.59 -462.03 -81.98 -58.04 22.38`
	`filterror =  0.02  0.99  1.95  1.97  2.00  2.01  2.03  2.04`
	
	These eight samples correspond to the value of the reach field, which shows whether the local client received the last eight NTP packets.

show ntp status

This is an example of output from the show ntp status command:

```
USSP-B33S-SW01#sho ntp status
Clock is synchronized, stratum 2, reference is 10.4.2.254
nominal freq is 250.0000 Hz, actual freq is 250.5630 Hz, precision is 2**18
reference time is D36968F7.7E3019A9 (02:12:07.492 UTC Fri May 25 2012)
clock offset is 417.2868 msec, root delay is 2.85 msec
root dispersion is 673.42 msec, peer dispersion is 261.80 msec
```

Terms already defined in the show up association section or the show ntp association detail section are not repeated.

Term	Explanation
precision	Precision is determined automatically and is measured as a power of two. In the example, 2**18 means 2^(-18), or 3.8 microseconds. Loss of synchronization between NTP peers or between a primary and client can be due to a variety of causes. NTP avoids synchronization with a machine whose time can be ambiguous in these ways: 1. NTP never synchronizes to a machine that is not synchronized itself. 1. NTP compares the time that is reported by several machines and does not synchronize to a machine whose time is significantly different from the others, even if its stratum is lower.

Troubleshoot NTP with Debugs

Some of the most common causes of NTP issues are:

- NTP packets are not received.
- NTP packets are received, but are not processed by the NTP process on the Cisco IOS.
- NTP packets are processed, but erroneous factors or packet data causes the loss of synchronization.
- NTP clock-period is manually set.

Important debug commands that help isolate the cause of these issues include:

- debug ip packets <acl>
- debug ntp packets
- debug ntp validity
- debug ntp sync
- debug ntp events

The next sections illustrate the use of debugs in order to resolve these common issues.

Note: Use the Command Lookup Tool in order to obtain more information on the commands used in this section. Only registered Cisco users can access internal tools and information.

Note: Refer to Important Information on Debug Commands before you use debug commands.

NTP Packets Not Received

Use the debug ip packet command in order to check if NTP packets are received and sent. Since debug output can be chatty, you can limit debug output with the use of Access Control Lists (ACLs). NTP uses User Datagram Protocol (UDP) port 123.

1. Create ACL 101:

```
access-list 101 permit udp any any eq 123
access-list 101 permit udp any eq 123 any
```

NTP packets usually have a source and destination port of 123, so this helps:

```
permit udp any eq 123 any eq 123
```

2. Use this ACL in order to limit output from the debug ip packet command:

```
debug ip packet 101
```

3. If the issue is with particular peers, narrow the ACL 101 down to those peers. If the peer is 172.16.1.1, change ACL 101 to:

```
access-list 101 permit udp host 172.16.1.1 any eq 123
access-list 101 permit udp any eq 123 host 172.16.1.1
```

This example output indicates that packets are not sent:

```
241925: Apr 23 2012 15:46:26.101 ETE: IP: s=10.50.38.70 (Tunnel99), d=10.50.44.101, len 76, input featu
241926: Apr 23 2012 15:46:26.101 ETE:     UDP src=123, dst=123, Ingress-NetFlow(13), rtype 0, forus FAL
sendself FALSE, mtu 0
241927: Apr 23 2012 15:46:26.101 ETE: IP: s=10.50.38.70 (Tunnel99), d=10.50.44.101, len 76, input featu
241928: Apr 23 2012 15:46:26.101 ETE:     UDP src=123, dst=123, MCI Check(55), rtype 0, forus FALSE,
sendself FALSE, mtu 0
```

Once you confirm that NTP packets are not received, you must:

- Check if NTP is configured correctly.
- Check if an ACL blocks NTP packets.
- Check for routing issues to the source or destination IP.

NTP Packets Not Processed

With both debug ip packet and debug ntp packets commands enabled, you can see the packets that are received and transmitted, and you can see that NTP acts on those packets. For every NTP packet received (as shown by debug ip packet), there is a correspondent entry generated by debug ntp packets .

This is the debug output when the NTP process works on received packets:

```
Apr 20 00:16:34.143 UTC: IP: tableid=0, s=10.3.2.31 (local), d=10.1.2.254 (Vlan2), routed via FIB
.Apr 20 00:16:34.143 UTC: IP: s=10.3.2.31 (local), d=10.1.2.254 (Vlan2), len 76, sending
.Apr 20 00:16:34.143 UTC: IP: s=10.3.2.31 (local), d=10.1.2.254 (Vlan2), len 76, sending full packet
.Apr 20 00:16:34.143 UTC: NTP: xmit packet to 10.1.2.254:
.Apr 20 00:16:34.143 UTC:  leap 3, mode 3, version 3, stratum 0, ppoll 64
.Apr 20 00:16:34.143 UTC:  rtdel 0021 (0.504), rtdsp 1105E7 (17023.056), refid 0A0102FE (10.1.2.254)
```

```
.Apr 20 00:16:34.143 UTC:  ref D33B2922.24FEBDC7 (00:15:30.144 UTC Fri Apr 20 2012)
.Apr 20 00:16:34.143 UTC:  org 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
.Apr 20 00:16:34.143 UTC:  rec 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
.Apr 20 00:16:34.143 UTC:  xmt D33B2962.24CAFAD1 (00:16:34.143 UTC Fri Apr 20 2012)
.Apr 20 00:16:34.143 UTC: IP: s=10.1.2.254 (Vlan2), d=10.3.2.31, len 76, rcvd 2
.Apr 20 00:16:34.143 UTC: NTP: rcv packet from 10.1.2.254 to 10.3.2.31 on Vlan2:
.Apr 20 00:16:34.143 UTC:  leap 0, mode 4, version 3, stratum 1, ppoll 64
.Apr 20 00:16:34.143 UTC:  rtdel 0000 (0.000), rtdsp 009D (2.396), refid 47505300 (10.80.83.0)
.Apr 20 00:16:34.143 UTC:  ref D33B2952.4CC11CCF (00:16:18.299 UTC Fri Apr 20 2012)
.Apr 20 00:16:34.143 UTC:  org D33B2962.24CAFAD1 (00:16:34.143 UTC Fri Apr 20 2012)
.Apr 20 00:16:34.143 UTC:  rec D33B2962.49D3724D (00:16:34.288 UTC Fri Apr 20 2012)
.Apr 20 00:16:34.143 UTC:  xmt D33B2962.49D997D0 (00:16:34.288 UTC Fri Apr 20 2012)
.Apr 20 00:16:34.143 UTC:  inp D33B2962.25010310 (00:16:34.144 UTC Fri Apr 20 2012)
.Apr 20 00:16:36.283 UTC: IP: tableid=0, s=10.3.2.31 (local), d=10.8.2.254 (Vlan2), routed via FIB
.Apr 20 00:16:36.283 UTC: IP: s=10.3.2.31 (local), d=10.8.2.254 (Vlan2), len 76, sending
.Apr 20 00:16:36.283 UTC: IP: s=10.3.2.31 (local), d=10.8.2.254 (Vlan2), len 76, sending full packet
.Apr 20 00:16:36.283 UTC: NTP: xmit packet to 10.8.2.254:
.Apr 20 00:16:36.283 UTC:  leap 3, mode 3, version 3, stratum 0, ppoll 64
.Apr 20 00:16:36.283 UTC:  rtdel 002F (0.717), rtdsp 11058F (17021.713), refid 0A0102FE (10.1.2.254)
.Apr 20 00:16:36.283 UTC:  ref D33B2962.25010310 (00:16:34.144 UTC Fri Apr 20 2012)
.Apr 20 00:16:36.283 UTC:  org 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
.Apr 20 00:16:36.283 UTC:  rec 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
.Apr 20 00:16:36.283 UTC:  xmt D33B2964.48947E87 (00:16:36.283 UTC Fri Apr 20 2012)
.Apr 20 00:16:36.283 UTC: IP: s=10.8.2.254 (Vlan2), d=10.3.2.31, len 76, rcvd 2
.Apr 20 00:16:36.283 UTC: NTP: rcv packet from 10.8.2.254 to 10.3.2.31 on Vlan2:
.Apr 20 00:16:36.283 UTC:  leap 0, mode 4, version 3, stratum 1, ppoll 64
.Apr 20 00:16:36.283 UTC:  rtdel 0000 (0.000), rtdsp 0017 (0.351), refid 47505300 (10.80.83.0)
.Apr 20 00:16:36.283 UTC:  ref D33B295B.8AF7FE33 (00:16:27.542 UTC Fri Apr 20 2012)
.Apr 20 00:16:36.283 UTC:  org D33B2964.48947E87 (00:16:36.283 UTC Fri Apr 20 2012)
.Apr 20 00:16:36.283 UTC:  rec D33B2964.4A6AD269 (00:16:36.290 UTC Fri Apr 20 2012)
.Apr 20 00:16:36.283 UTC:  xmt D33B2964.4A7C00D0 (00:16:36.290 UTC Fri Apr 20 2012)
.Apr 20 00:16:36.283 UTC:  inp D33B2964.498A755D (00:16:36.287 UTC Fri Apr 20 2012)
```

This is an example where NTP does not work on received packets. Although NTP packets are received (as shown by debug ip packets), the NTP process does not act on them. For NTP packets that are sent out, a corresponding debug ntp packets output is present, because the NTP process has to generate the packet. The issue is specific to received NTP packets that are not processed.

```
071564: Apr 23 2012 15:46:26.100 ETE: NTP: xmit packet to 10.50.44.101:
071565: Apr 23 2012 15:46:26.100 ETE:  leap 0, mode 1, version 3, stratum 5, ppoll 1024
071566: Apr 23 2012 15:46:26.100 ETE:  rtdel 07B5 (30.106), rtdsp 0855 (32.547), refid 0A32266A
(10.50.38.106)
071567: Apr 23 2012 15:46:26.100 ETE:  ref D33FDB05.1A084831 (15:43:33.101 ETE Mon Apr 23 2012)
071568: Apr 23 2012 15:46:26.100 ETE:  org 00000000.00000000 (01:00:00.000 HIVER Mon Jan 1 1900)
071569: Apr 23 2012 15:46:26.100 ETE:  rec 00000000.00000000 (01:00:00.000 HIVER Mon Jan 1 1900)
071570: Apr 23 2012 15:46:26.100 ETE:  xmt D33FDBB2.19D3457C (15:46:26.100 ETE Mon Apr 23 2012)
PCY_PAS1#
071571: Apr 23 2012 15:47:31.497 ETE: IP: s=10.50.38.78 (Tunnel99), d=10.50.44.69, len 76, input featur
071572: Apr 23 2012 15:47:31.497 ETE:     UDP src=123, dst=123, Ingress-NetFlow(13), rtype 0, forus FAL
sendself FALSE, mtu 0
071573: Apr 23 2012 15:47:31.497 ETE: IP: s=10.50.38.78 (Tunnel99), d=10.50.44.69, len 76, input featur
071574: Apr 23 2012 15:47:31.497 ETE:     UDP src=123, dst=123, MCI Check(55), rtype 0, forus FALSE,
sendself FALSE, mtu 0
071575: Apr 23 2012 15:47:31.497 ETE: FIBipv4-packet-proc: route packet from Tunnel99 src 10.50.38.78 d
10.50.44.69
071576: Apr 23 2012 15:47:31.497 ETE: FIBfwd-proc: base:10.50.44.69/32 receive entry
PCY_PAS1#
071577: Apr 23 2012 15:47:31.497 ETE: FIBipv4-packet-proc: packet routing failed
```

```
071578: Apr 23 2012 15:47:31.497 ETE: IP: s=10.50.38.78 (Tunnel99), d=10.50.44.69, len 76, rcvd 2
071579: Apr 23 2012 15:47:31.497 ETE:     UDP src=123, dst=123
071580: Apr 23 2012 15:47:31.497 ETE: IP: s=10.50.38.78 (Tunnel99), d=10.50.44.69, len 76, stop process
for forus packet
071581: Apr 23 2012 15:47:31.497 ETE:     UDP src=123, dst=123
PCY_PAS1#
071582: Apr 23 2012 16:03:30.105 ETE: NTP: xmit packet to 10.50.44.101:
071583: Apr 23 2012 16:03:30.105 ETE:  leap 0, mode 1, version 3, stratum 5, ppoll 1024
071584: Apr 23 2012 16:03:30.105 ETE:  rtdel 0759 (28.702), rtdsp 087D (33.157), refid 0A32266A
(10.50.38.106)
071585: Apr 23 2012 16:03:30.105 ETE:  ref D33FDF05.1B2CC3D4 (16:00:37.106 ETE Mon Apr 23 2012)
071586: Apr 23 2012 16:03:30.105 ETE:  org 00000000.00000000 (01:00:00.000 HIVER Mon Jan 1 1900)
071587: Apr 23 2012 16:03:30.105 ETE:  rec 00000000.00000000 (01:00:00.000 HIVER Mon Jan 1 1900)
071588: Apr 23 2012 16:03:30.105 ETE:  xmt D33FDFB2.1B1D5E7E (16:03:30.105 ETE Mon Apr 23 2012)
PCY_PAS1#
071589: Apr 23 2012 16:04:35.502 ETE: IP: s=10.50.38.78 (Tunnel99), d=10.50.44.69, len 76, input featur
071590: Apr 23 2012 16:04:35.506 ETE:     UDP src=123, dst=123, Ingress-NetFlow(13), rtype 0, forus FAL
sendself FALSE, mtu 0
071591: Apr 23 2012 16:04:35.506 ETE: IP: s=10.50.38.78 (Tunnel99), d=10.50.44.69, len 76, input featur
071592: Apr 23 2012 16:04:35.506 ETE:     UDP src=123, dst=123, MCI Check(55), rtype 0, forus FALSE,
sendself FALSE, mtu 0
071593: Apr 23 2012 16:04:35.506 ETE: FIBipv4-packet-proc: route packet from Tunnel99 src 10.50.38.78 d
10.50.44.69
071594: Apr 23 2012 16:04:35.506 ETE: FIBfwd-proc: base:10.50.44.69/32 receive entry
PCY_PAS1#
071595: Apr 23 2012 16:04:35.506 ETE: FIBipv4-packet-proc: packet routing failed
071596: Apr 23 2012 16:04:35.506 ETE: IP: s=10.50.38.78 (Tunnel99), d=10.50.44.69, len 76, rcvd 2
071597: Apr 23 2012 16:04:35.506 ETE:     UDP src=123, dst=123
071598: Apr 23 2012 16:04:35.506 ETE: IP: s=10.50.38.78 (Tunnel99), d=10.50.44.69, len 76, stop process
for forus packet
071599: Apr 23 2012 16:04:35.506 ETE:     UDP src=123, dst=123
PCY_PAS1#
```

Loss of Synchronization

Loss of synchronization can occur if the dispersion and/or delay value for a server goes very high. High values indicate that the packets take too long to get to the client from the server/peer in reference to the root of the clock. So, the local machine cannot trust the accuracy of the time present in the packet, because it does not know how long it took for the packet to get here.

NTP is meticulous about time and can not synchronize with another device it cannot trust or cannot adjust in a way so that it can be trusted.

If there is a saturated link and buffering occurs along the way, the packets get delayed as they come to the NTP client. So, the timestamp contained in a subsequent NTP packet can occasionally vary a lot, and the local client cannot really adjust for that variance.

NTP does not offer a method to turn off the validation of these packets unless you use SNTP (Simple Network Time Protocol). SNTP is not much of an alternative because it is not widely supported in software.

If you experience loss of synchronization, you must check the links:

- Are they saturated?
- Are there any kinds of drops in your wide-area network (WAN) links
- Does encryption occur?

Monitor the reach value from the show ntp associations detail command. The highest value is 377. If the

value is 0 or low, NTP packets are received intermittently, and the local client goes out of sync with the server.

debug ntp validity

The debug ntp validity command indicates whether the NTP packet failed sanity or validity checks and reveals the reason for the failure. Compare this output to the sanity tests specified in RFC1305 that are used in order to test the NTP packet received from a server. Eight tests are defined:

Test	Mask	Explanation
1	0x01	Duplicate packet received
2	0x02	Bogus packet received
3	0x04	Protocol unsynchronized
4	0x08	Peer delay/dispersion failed boundary check
5	0x10	Peer authentication failed
6	0x20	Peer clock unsynchronized (common for unsynched server)
7	0x40	Peer stratum out of bound
8	0x80	Root delay/dispersion failed boundary check

This is sample output of from the debug ntp validity command:

```
PCY_PAS1#debug ntp validity
NTP peer validity debugging is on

009585: Mar  1 2012 09:14:32.670 HIVER: NTP: packet from 192.168.113.57 failed validity tests 52
009586: Mar  1 2012 09:14:32.670 HIVER: Authentication failed
009587: Mar  1 2012 09:14:32.670 HIVER: Peer/Server Stratum out of bound
PCY_PAS1#
009588: Mar  1 2012 09:14:38.210 HIVER: NTP: packet from 192.168.56.1 failed validity tests 14
009589: Mar  1 2012 09:14:38.210 HIVER: Authentication failed
PCY_PAS1#
009590: Mar  1 2012 09:14:43.606 HIVER: NTP: packet from 10.110.103.27 failed validity tests 14
009591: Mar  1 2012 09:14:43.606 HIVER: Authentication failed
PCY_PAS1#
009592: Mar  1 2012 09:14:48.686 HIVER: NTP: packet from 192.168.113.57failed validity tests 52
009593: Mar  1 2012 09:14:48.686 HIVER: Authentication failed
009594: Mar  1 2012 09:14:48.686 HIVER: Peer/Server Stratum out of bound
```

```
PCY_PAS1#
009596: Mar  1 2012 09:14:54.222 HIVER: NTP: packet from 10.110.103.35 failed validity tests 14
009597: Mar  1 2012 09:14:54.222 HIVER: Authentication failed
PCY_PAS1#
009598: Mar  1 2012 09:14:54.886 HIVER: NTP: synced to new peer 10.50.38.106
009599: Mar  1 2012 09:14:54.886 HIVER: NTP: 10.50.38.106 synced to new peer
PCY_PAS1#
009600: Mar  1 2012 09:14:59.606 HIVER: NTP: packet from 10.110.103.27 failed validity tests 14
009601: Mar  1 2012 09:14:59.606 HIVER: Authentication failed
PCY_PAS1#
009602: Mar  1 2012 09:15:04.622 HIVER: NTP: packet from 192.168.113.137 failed validity tests 52
009603: Mar  1 2012 09:15:04.622 HIVER: Authentication failed
009604: Mar  1 2012 09:15:04.622 HIVER: Peer/Server Stratum out of bound
PCY_PAS1#
009605: Mar  1 2012 09:15:10.238 HIVER: NTP: packet from 192.168.56.1 failed validity tests 14
009606: Mar  1 2012 09:15:10.238 HIVER: Authentication failed
PCY_PAS1#
009607: Mar  1 2012 09:15:15.338 HIVER: NTP: packet from 10.83.23.140 failed validity tests 52
009608: Mar  1 2012 09:15:15.338 HIVER: Authentication failed
009609: Mar  1 2012 09:15:15.338 HIVER: Peer/Server Stratum out of bound
PCY_PAS1#
009610: Mar  1 2012 09:15:20.402 HIVER: NTP: packet from 192.168.113.92 failed validity tests 74
009611: Mar  1 2012 09:15:20.402 HIVER: Authentication failed
009612: Mar  1 2012 09:15:20.402 HIVER: Peer/Server Clock unsynchronized
009613: Mar  1 2012 09:15:20.402 HIVER: Peer/Server Stratum out of bound
```

debug ntp packets

You can use the debug ntp packets command in order to see the time that the peer/server gives you in the received packet. The time local machine also tells the time it knows to the peer/server in the transmitted packet.

Field	rcv Packet	xmit Packet
org	Originator time stamp, which is the server time.	Originator (client) time stamp when it sent the packet. (Client originates a packet to the server.)
rec	Time stamp on the client when it received the packet.	Client current time.

In this sample output, the time stamps in the received packet from the server and the packet sent to another server are the same, which indicates that the client NTP is in sync.

```
USSP-B33S-SW01#debug ntp packets
NTP packets debugging is on
USSP-B33S-SW01#
May 25 02:21:48.182 UTC: NTP: rcv packet from 10.1.2.254 to 10.3.2.31 on Vlan2:
May 25 02:21:48.182 UTC:  leap 0, mode 4, version 3, stratum 1, ppoll 64
May 25 02:21:48.182 UTC:  rtdel 0000 (0.000), rtdsp 00F2 (3.693), refid 47505300 (10.80.83.0)
May 25 02:21:48.182 UTC:  ref D3696B38.B722C417 (02:21:44.715 UTC Fri May 25 2012)
May 25 02:21:48.182 UTC:  org D3696B3C.2EA179BA (02:21:48.182 UTC Fri May 25 2012)
May 25 02:21:48.182 UTC:  rec D3696B3D.E58DE1BE (02:21:49.896 UTC Fri May 25 2012)
May 25 02:21:48.182 UTC:  xmt D3696B3D.E594E7AF (02:21:49.896 UTC Fri May 25 2012)
May 25 02:21:48.182 UTC:  inp D3696B3C.2EDFC333 (02:21:48.183 UTC Fri May 25 2012)
May 25 02:22:46.051 UTC: NTP: xmit packet to 10.4.2.254:
May 25 02:22:46.051 UTC:  leap 0, mode 3, version 3, stratum 2, ppoll 64
May 25 02:22:46.051 UTC:  rtdel 00C0 (2.930), rtdsp 1C6FA (1777.252), refid 0A0402FE (10.4.2.254)
May 25 02:22:46.051 UTC:  ref D3696B36.33D43F44 (02:21:42.202 UTC Fri May 25 2012)
```

```
May 25 02:22:46.051 UTC:  org D3696B37.E72C75AE (02:21:43.903 UTC Fri May 25 2012)
May 25 02:22:46.051 UTC:  rec D3696B36.33D43F44 (02:21:42.202 UTC Fri May 25 2012)
May 25 02:22:46.051 UTC:  xmt D3696B76.0D43AE7D (02:22:46.051 UTC Fri May 25 2012)
```

This is an example of output when the clocks are not in sync. Notice the time difference between the xmit packet and the rcv packet. The peer dispersion can be at the max value of 16000, and the reach for the peer can show 0.

```
USSP-B33S-SW01#
.May 25 02:05:59.011 UTC: NTP: xmit packet to 10.4.2.254:
.May 25 02:05:59.011 UTC:  leap 3, mode 3, version 3, stratum 0, ppoll 64
.May 25 02:05:59.011 UTC:  rtdel 00A3 (2.487), rtdsp 1104D0 (17018.799), refid 0A0402FE (10.4.2.254)
.May 25 02:05:59.011 UTC:  ref D3696747.03D8661A (02:04:55.015 UTC Fri May 25 2012)
.May 25 02:05:59.011 UTC:  org 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
.May 25 02:05:59.011 UTC:  rec 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
.May 25 02:05:59.011 UTC:  xmt D3696787.03105783 (02:05:59.011 UTC Fri May 25 2012)
.May 25 02:05:59.011 UTC: NTP: rcv packet from 10.4.2.254 to 10.3.2.31 on Vlan2:
.May 25 02:05:59.011 UTC:  leap 0, mode 4, version 3, stratum 1, ppoll 64
.May 25 02:05:59.011 UTC:  rtdel 0000 (0.000), rtdsp 0014 (0.305), refid 47505300 (10.80.83.0)
.May 25 02:05:59.011 UTC:  ref D3696782.C96FD778 (02:05:54.786 UTC Fri May 25 2012)
.May 25 02:05:59.011 UTC:  org D3696787.03105783 (02:05:59.011 UTC Fri May 25 2012)
.May 25 02:05:59.011 UTC:  rec D3696787.281A963F (02:05:59.156 UTC Fri May 25 2012)
.May 25 02:05:59.011 UTC:  xmt D3696787.282832C4 (02:05:59.156 UTC Fri May 25 2012)
.May 25 02:05:59.011 UTC:  inp D3696787.03C63542 (02:05:59.014 UTC Fri May 25 2012)
```

debug ntp sync and debug ntp events

The debug ntp sync command produces one-line outputs that show whether the clock has synced or the sync has changed. The command is generally enabled with debug ntp events.

The debug ntp events command shows any NTP events that occur, which helps you determine if a change in the NTP triggered an issue such as clocks that go out of sync. (In other words, if your happily synced clocks suddenly go crazy, you know to look for a change or trigger!)

This is an example of both debugs. Initially, the client clocks were synced. The debug ntp events command shows that an NTP peer stratum change occurred, and the clocks then went out of sync.

```
USSP-B33S-SW01#debug ntp sync
NTP clock synchronization debugging is on
USSP-B33S-SW01#
USSP-B33S-SW01#
USSP-B33S-SW01#debug ntp events
NTP events debugging is on
USSP-B33S-SW01#
USSP-B33S-SW01#
May 25 02:25:57.620 UTC: NTP: xmit packet to 10.4.2.254:
May 25 02:25:57.620 UTC:  leap 0, mode 3, version 3, stratum 2, ppoll 64
May 25 02:25:57.620 UTC:  rtdel 00D4 (3.235), rtdsp 26B26 (2418.549), refid 0A0402FE (10.4.2.254)
May 25 02:25:57.620 UTC:  ref D3696BF5.C47EB880 (02:24:53.767 UTC Fri May 25 2012)
May 25 02:25:57.620 UTC:  org D3696BF7.E5F91077 (02:24:55.898 UTC Fri May 25 2012)
May 25 02:25:57.620 UTC:  rec D3696BF5.C47EB880 (02:24:53.767 UTC Fri May 25 2012)
May 25 02:25:57.620 UTC:  xmt D3696C35.9ED1CE97 (02:25:57.620 UTC Fri May 25 2012)
```

```
May 25 02:25:57.620 UTC: NTP: rcv packet from 10.4.2.254 to 10.3.2.31 on Vlan2:
May 25 02:25:57.620 UTC:  leap 0, mode 4, version 3, stratum 1, ppoll 64
May 25 02:25:57.620 UTC:  rtdel 0000 (0.000), rtdsp 000E (0.214), refid 47505300 (10.80.83.0)
May 25 02:25:57.620 UTC:  ref D3696C37.D528800E (02:25:59.832 UTC Fri May 25 2012)
May 25 02:25:57.620 UTC:  org D3696C35.9ED1CE97 (02:25:57.620 UTC Fri May 25 2012)
May 25 02:25:57.620 UTC:  rec D3696C37.E5C7AB3D (02:25:59.897 UTC Fri May 25 2012)
May 25 02:25:57.620 UTC:  xmt D3696C37.E5D1F273 (02:25:59.897 UTC Fri May 25 2012)
May 25 02:25:57.620 UTC:  inp D3696C35.9F9EA2C4 (02:25:57.623 UTC Fri May 25 2012)
May 25 02:25:59.830 UTC: NTP: peer stratum change
May 25 02:25:59.830 UTC: NTP: clock reset
May 25 02:25:59.830 UTC: NTP: sync change
May 25 02:25:59.830 UTC: NTP: peer stratum change
May 25 02:26:05.817 UTC: NTP: xmit packet to 10.1.2.254:
May 25 02:26:05.817 UTC:  leap 3, mode 3, version 3, stratum 0, ppoll 64
May 25 02:26:05.817 UTC:  rtdel 00C2 (2.960), rtdsp 38E9C (3557.068), refid 0A0402FE (10.4.2.254)
May 25 02:26:05.817 UTC:  ref D3696C35.9F9EA2C4 (02:25:57.623 UTC Fri May 25 2012)
May 25 02:26:05.817 UTC:  org 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
May 25 02:26:05.817 UTC:  rec 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
May 25 02:26:05.817 UTC:  xmt D3696C3D.D12D0565 (02:26:05.817 UTC Fri May 25 2012)
```

NTP Clock-period Manually Set

The Cisco.com website warns that:

> "The ntp clock-period command is automatically generated to reflect the correction factor that constantly changes when the copy running-configuration startup-configuration command is entered to save the configuration to NVRAM. Do not attempt to manually use the ntp clock-period command. Ensure that you remove this command line when you copy configuration files to other devices."

The clock-period value is dependent on the hardware, so it differs for every device.

The ntp clock-period command automatically appears in the configuration when you enable NTP. The command is used in order to adjust the software clock. The 'adjustment value' compensates for the 4 msec tick interval, so that, with the minor adjustment, you have 1 second at the end of the interval.

If the device has calculated that its system clock loses time (perhaps there needs to be a frequency compensation from the base level of the router), it automatically adds this value to the system clock in order to maintain its synchronicity.

Note: This command must not be changed by the user.

The default NTP clock-period for a router is 17179869 and is essentially used in order to start the NTP process.

The conversion formula is 17179869 * 2^(-32) = 0.003999999957159161567687988281 25, or approximately 4 milliseconds.

For example, the system clock for the Cisco 2611 routers (one of the Cisco 2600 Series Routers) was found to be slightly out-of-sync and could be resynchronized with this command:

```
ntp clock-period 17208078
```

This equals 17208078 * 2^(-32) = 0.004006567876785993576049804687 5, or a little over 4 milliseconds.

Cisco recommends that you let the router run for a week or so in normal network conditions and then use the wr mem command in order to save the value. This gives you an accurate figure for next reboot and allows NTP to synchronize more quickly.

Use the no ntp clock-period command when you save the configuration for use on another device because this command drops the clock-period back to the default of that particular device. You can recalcuate the true value (but can reduce the accuracy of the system clock during that recalculation time period).

Remember that this value is hardware dependent, so if you copy a configuration and use it on different devices, you can cause problems. Cisco plans to replace NTP version 3 with version 4 in order to resolve this issue.

If you are not aware of these issues, you can decide to manually tinker with this value. In order to migrate from one device to another, you can decide to copy the old configuration and paste it on the new device. Unfortunately, because the ntp clock-period command appears in the running-config and startup-config, NTP clock-period is pasted on the new device. When this happens, NTP on the new client always goes out of sync with the server with a high peer dispersion value.

Instead, clear the NTP clock-period with the no ntp clock-period command, then save the configuration. The router eventually calculates a clock-period appropriate for itself.

The ntp clock-period command is no longer available in Cisco IOS software Version 15.0 or later; the parser now rejects the command with the error:

```
"%NTP: This configuration command is deprecated."
```

You are not allowed to configure the clock-period manually, and the clock-period is not allowed in the running-config. Since the parser rejects the command if it was in the start-up config (in earlier Cisco IOS versions such as 12.4), the parser rejects the command when it copies the start-up config to the running-config on boot-up.

The new, replacement command is ntp clear drift.

Related Information

- Support Forum Thread: NTP clock-period not configured
- Network Time Protocol: Best Practices White Paper
- Troubleshoot Network Time Protocol (NTP)
- Cisco Technical Support & Downloads