



## The Windows Time Service

*By Shala Brandolini and Darin Green, Microsoft Corporation*

*Published: April 2001*

---

### **Abstract**

The Windows® 2000 operating system implements Kerberos V5 as the primary protocol for network authentication. One requirement of the protocol is that system clocks must be synchronized. To achieve the degree of clock synchronization that Kerberos authentication requires, Windows 2000 implements the Windows Time service (or W32Time). This paper describes how W32Time is designed and explains how to configure and troubleshoot the service.

*The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.*

*This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.*

*Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.*

*Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.*

*© 2001 Microsoft Corporation. All rights reserved. Microsoft, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.*

*Other product and company names mentioned herein may be the trademarks of their respective owners.*

*Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA*

*4/2001*

---

## Contents

<b>Introduction</b> .....	<b>1</b>
<b>Time and the Kerberos Authentication Protocol</b> .....	<b>2</b>
<b>Potential Security Vulnerabilities</b> .....	<b>3</b>
Ticket Acceptance Attacks .....	3
Replay Attacks .....	4
Denial of Service Attacks .....	4
<b>Time Protocols</b> .....	<b>6</b>
Network Time Protocol .....	6
Simple Network Time Protocol .....	6
SNTP Security .....	7
<b>Windows Time Service</b> .....	<b>8</b>
Time Convergence .....	8
Time Convergence Hierarchy .....	9
Domain Hierarchy Based Synchronization .....	10
Manually Specified Synchronization .....	13
No Specified Synchronization .....	14
<b>W32Time Service Tools</b> .....	<b>15</b>
Net Time .....	15
W32tm .....	16
<b>Configuring W32Time</b> .....	<b>18</b>
W32Time Registry Entries .....	18
Designating an External Time Source .....	19
Manually Starting and Stopping W32Time .....	20
Adjusting the Kerberos Time Skew Variable .....	20
<b>Troubleshooting Time-Related Errors</b> .....	<b>21</b>
Unable to Locate Time Server .....	21
Access Denied .....	21
Failure to Bind .....	22
Unable to Log On to the Network .....	22

<b>FAQ</b> .....	<b>23</b>
<b>Appendix A: Microsoft Windows IP Security</b> .....	<b>25</b>
Creating IPsec Policies .....	25
<b>Related Links</b> .....	<b>28</b>

---

## Introduction

Time synchronization is important in a Windows® 2000 environment because Windows 2000 implements the Kerberos V5 authentication protocol, a standards-based authentication protocol defined by RFC 1510. W32Time meets the requirements specified by the Kerberos authentication protocol to provide clock values that are "loosely synchronized" across a network. This service is not designed for use by applications that require greater precision.

This document provides a detailed description of the Windows Time service, how it operates on a Windows 2000 network, and how it can be configured to best meet the needs of your enterprise. It also covers issues involving time service security.

---

## Time and the Kerberos Authentication Protocol

Windows 2000 uses Kerberos V5 as the primary method of authenticating users and computers in a network domain. Details of the Kerberos authentication protocol are beyond the scope of this paper. For our purpose here, what is important to know about the Kerberos protocol is that a Kerberos client proves its identity to a server by presenting a ticket. Since only parts of the ticket are encrypted, (some parts are encrypted, but this encryption doesn't thwart replay) and might be intercepted and reused by an attacker, additional information is sent with the ticket to prove the identity of the client. This information (called the authenticator) is encrypted by the client with a secret key, and includes a timestamp. The timestamp proves that the message was recently generated and is not a replay.

When the server receives the ticket, it decrypts the authenticator and extracts the client's clock time. The server then performs the following checks before acknowledging the client:

- Checks the client's time to make sure that it falls within the server's time and the allowable skew.
- Checks the client's time to make sure that the time is not the same as or earlier than the time of another authenticator.

If the client's time falls within the allowable skew and its timestamp is unique, the server then slightly modifies the contents of the original authenticator and re-encrypts it with the client's secret key, establishing mutual authentication. The server then acknowledges the client by sending the modified authenticator with the client's original timestamp back to the client for identification.

For a more detailed description of Microsoft's implementation of the Kerberos V5 authentication protocol, see [Windows 2000 Kerberos Authentication](#) on the Windows 2000 Web site.

---

## Potential Security Vulnerabilities

Tickets presented by Kerberos clients include a timestamp in order to prevent attacks by malicious users. By modifying or spoofing the intended time source, a malicious individual might attempt to compromise an enterprise in one of the following three ways:

- **Ticket Acceptance** – A malicious user attempts to cause the time server to accept a Kerberos ticket that has expired. Kerberos tickets rely on time for validation and expire after a period of time.
- **Replay Attacks** – A malicious user captures an authentication request, for which he/she has managed to determine the session key, and replays it in order to gain access to an otherwise restricted resource.
- **Denial of Service** – A malicious user continually skews the time of the time source to produce an excessive amount of time resynchronization traffic, eventually causing a denial of service.

Windows 2000 can defend against each of these types of attacks. The methods for defense are listed below:

### Ticket Acceptance Attacks

Windows 2000 Professional computers, and stand-alone and member servers get their time from a domain controller in the domain to which they are joined. When the server receives the ticket, it decrypts the authenticator and extracts the client's clock time. Before acknowledging the client, the server checks the client's time to make sure that it falls within the server's allowable time skew. If the client's time is not within the allowable skew, the server returns the following error:

```
KRB_AP_ERR_SKEW 0x25          "Clock skew too great"
```

The client compensates for the skew error by using the time returned in the error. A time skew between the client and server does not cause the client to change its system time, only the time that is used to communicate with that server. This can only work if the skew is within the lifetime of a ticket. The client uses this server skew value until it detects another time skew error (usually when the time is changed on any server that the client is communicating). When a computer detects that its time is skewed relative to its domain, it will call on the Windows Time service to correct the skew by synchronizing with a domain controller. It is possible for multiple trusted domains (and forests) to have different times. By using skew correction, clients can successfully authenticate to servers in domains with different times without having to resynchronize the client clock.

If the time is within the allowable *skew*, the server accepts the authentication request from that client. By default, the allowable skew is set at five minutes, but this can be modified to allow a larger or smaller window of time for the exchange of authenticators between client and server. (For details about how to change the Kerberos skew setting and reasons for doing so, see "Configuring W32Time" later in this article.)

---

**Note After receiving a clock skew error, a client is allowed to re-attempt authentication up to four times before Kerberos forces the client's clock to synchronize with its own.**

---

A Windows 2000 domain controller attempting to authenticate to another Windows 2000 domain controller attempts to thwart a ticket acceptance attack the same way that a Windows 2000 Professional computer, or

stand-alone or member server does. Domain controllers communicate with each other only for replication. When replication occurs, one domain controller acts as the client and the other acts as the server.

The following technical paper describes this mechanism in detail: Don Davis, Daniel Geer, and Theodore Ts'o, "Kerberos With Clocks Adrift: History, Protocols, and Implementation"  
<http://world.std.com/~dtd/synch/synch.ps>

## Replay Attacks

The following terminology is necessary to understand how Windows 2000 defends against replay attacks:

*security context* The security attributes or rules that are currently in effect. For SSPI, a *security context* is an opaque data structure that contains security data relevant to a connection, such as a session key or an indication of the duration of the session.

*security protocol* A specification that defines security-related data objects and rules about how the objects are used to maintain security on a computer system.

*security support provider (SSP)* A dynamic-link library (DLL) that implements the SSPI by making one or more security packages available to applications. Each security package provides mappings between an application's SSPI function calls and an actual security model's functions. Security packages support security protocols such as Kerberos authentication and the Microsoft® LAN Manager. For more information about SSPI see MSDN.

When a server receives a client's ticket, it decrypts the authenticator and extracts the client's clock time. Before acknowledging the client, the server also checks the client's time to make sure that the time is not the same as or earlier than the time of another authenticator. The server keeps a record of the times stamped on each authenticator received within the allowable skew time. If the server receives any messages during that skew time with a timestamp that is the same as or earlier than the time of the last message from the client, it rejects those messages and returns the following error to the client:

KRB\_AP\_ERR\_REPEAT 0x22 "The request is a replay"

However, if the application developer has used the Security Support Provider Interface (SSPI) [EncryptMessage\(\)](#) or [MakeSignature\(\)](#) APIs to protect the data, the underlying security protocol (i.e., Kerberos) in Windows 2000 provides additional protection against replay attacks. The following two "detections" are enforced by using these SSPI APIs in the Kerberos context:

- Out-of-Sequence Detection – The Kerberos Out-of-Sequence flag checks the sequence numbers on each message to ensure that the messages arrive in the correct order.
- Replay Detection – With Kerberos Replay Detection, one side of the application checks to see whether it has received the same message previously. The initiator can specify that it wants messages to be checked against a message replay cache. This is used to determine if the message is a replay of a previous message.

It is important to note that some applications do not protect the data stream using the Kerberos **security context**. Instead, they use Kerberos only for authentication, which leaves them vulnerable.

## Denial of Service Attacks

Secure time synchronization occurs with no additional configuration if the time source is within the Windows 2000 domain (or forest) hierarchy. However, in some organizations, the time source is external to the

domain hierarchy. If you are operating in a Windows 2000 environment, it is possible to receive time from an external time source; however, if you choose to implement this configuration, it is imperative that it be secure and protected against compromise. You might use external time sources for the following Windows 2000 computers in an enterprise:

- The domain root (PDC emulator) – This is commonly the case when the entire forest relies on this single point for time synchronization.
- Child domains or workstations – You might use an external time source when an application requires more accurate time than what w32time's SNTP implementation can provide. For more information about the SNTP Protocol, see "Simple Network Time Protocol" later in this whitepaper.
- The computer is not a member of a domain.

#### Securing External Time Sources with IP Security (IPSec)

The best method for securing connections to external time servers is to apply the industry standard IPSec transport mode and using IKE Identity Protection Mode (Main mode/quick mode). For a description of IPSec and example policies that can be used to secure your environment, see Appendix A: Microsoft Windows IP Security.

---

## Time Protocols

The Kerberos V5 specification requires the use of a distributed time synchronization protocol. The protocol must be able to read a server clock, transmit the reading to one or more clients, and adjust each client clock as required.

How well a time protocol is able to synchronize time depends on the limits of the underlying computer hardware and operating system. For example, the clock on a Windows 2000 system ticks approximately once every 10 milliseconds; therefore, no matter what time protocol is used on a Windows 2000–based system, it will not be accurate to more than 10 milliseconds. The time synchronization protocol, along with factors such as clock frequency drift, clock resolution, and network delay, determines how closely the server clock and the client clock are synchronized. W32Time uses the Simple Network Time Protocol (SNTP), which is derived from the Network Time Protocol (NTP).

### Network Time Protocol

The Network Time Protocol, specified in RFC 1305, is the most widely used protocol for computer clock synchronization. NTP synchronizes the system time on a computer to that of a server that has been synchronized by a reference source such as a radio, satellite receiver, or modem. With the appropriate hardware, NTP is capable of achieving synchronization to within microseconds, depending on the synchronization source and the network paths.

NTP synchronization is part of a software package that includes a full suite of NTP options and algorithms, which are relatively complex, real-time applications. The sheer size and complexity of the NTP suite is not appropriate for many environments that do not have stringent accuracy requirements.

### Simple Network Time Protocol

The Simple Network Time Protocol, specified in RFC 1769, is a simplified access strategy for servers and clients that do not require the degree of accuracy that NTP provides. SNTP is designed to operate in a dedicated server configuration. With careful design and control of the various latencies that are typical in a dedicated network design, it is possible to deliver time accurate to the order of milliseconds by using SNTP.

SNTP is a basic version of NTP. Because the network packet formats of both protocols are identical, the two are interoperable. The main difference between the two protocols is that SNTP does not have the error management and complex filtering systems that NTP provides. The SNTP protocol allows room for error and might not be suitable for some corporations, such as those in the financial industry, that have very strict accuracy requirements. For this reason, RFC 1769 indicates "the use of SNTP rather than NTP in primary servers should be carefully considered."

Because the design goal behind W32Time was operability with Kerberos V5 authentication, and not keeping time accurate to the microsecond (as provided by the NTP protocol), developers chose to incorporate SNTP into the W32Time service running on primary servers and clients in Windows 2000. By using the SNTP protocol, W32Time meets the requirement of "loose synchronization" with other hosts on the network as specified in RFC 1510. The SNTP protocol ensures that all clocks in an enterprise are within 20 seconds of one another, and all clocks in a site are within two seconds of one another.

It is important to recognize that W32Time, run on its own, might not be sufficient for networks that require synchronization accuracy to within microseconds. SNTP is best for environments that do not need or cannot

justify a full NTP implementation and that can function when clocks are loosely synchronized. If your organization requires greater accuracy, you can designate reliable time sources throughout your domain against which computers can check their local time. (For information on designating reliable time sources, see "Time Convergence Hierarchy" later in this article.) If you require greater accuracy across a site, employ third-party software.

### **SNTP Security**

To ensure the security of the SNTP protocol, NTP authentication provisions have been added to the Windows 2000 implementation of SNTP. When a Windows 2000–based client that is a member of a domain logs on to the network, the Net Logon service provides a single access point to a domain controller. The client then provides its account password to its authenticating domain controller. Because the password is provided over a discreet communication channel, known as the Net Logon Secure Channel, only the domain controller and the client know the password. The secure channel is an encrypted connection that the Net Logon service established between the client and the nearest domain controller for its domain. W32Time uses the client's secure account password to generate a signature on SNTP packets that are sent across the network.

Because the Net Logon Secure Channel is used to set up and maintain the computer account password, the security of the time service is only as good as the security of the Net Logon Secure Channel, even though the secure channel is not directly involved in the activities of the time service. When a client requests the time from a domain controller in a domain hierarchy, the client requires that the time be authenticated. The domain controller then returns the required authentication in the form of a signed 64-bit hash of the time information. If the returned time is not signed or is signed incorrectly, the time is rejected. All such authentication failures are logged in the Event Log.

If a client is manually configured to access time from an NTP time server outside of the domain hierarchy, the SNTP packets sent between the client and the time server are not authenticated and therefore are not secure.

---

## Windows Time Service

W32Time is installed by default on all Windows 2000–based computers (Microsoft Windows 2000 Professional, Windows 2000 Server, Microsoft Windows 2000 Advanced Server). W32Time uses coordinated universal time (UTC), which is based on an atomic time scale and is the correct term to use when talking about time and time synchronization. UTC is the name for time that is independent of time zone. Time zone information is stored in the computer's registry and is added to the system time just before it is displayed to the user.

The W32Time service starts automatically on computers that are joined to a domain. For computers that are not joined to a domain, you can start the time service manually. (See "Configuring W32 Time" later in this article.)

On computers that are joined to a domain, time synchronization takes place when the W32Time service turns on during system startup. The Net Logon service looks for a domain controller that can authenticate and synchronize time with the client. When a domain controller is found, the client sends a request for time and waits for a reply from the domain controller. This communication is an exchange of SNTP packets intended to calculate the time offset and roundtrip delay between the two computers. (For complete information on SNTP packet structure, refer to RFC 1769).

### Time Convergence

Time convergence occurs throughout a network as each computer accesses time from a more reliable time server. A client issues a request to a time server for synchronization using the client computer's Relative ID (RID). The time server determines the secret password for the client's RID, and this password is renegotiated every 30 days between each computer and the Active Directory domain. The reliable server sends the current time the computer in the clear in the form of a signed SNTP packet. When the client receives the packet, it verifies its integrity and that the source of the packet is the requested time server. The information provided within the packet determines whether an adjustment needs to be made to the computer's current clock time so that it becomes synchronized with the more reliable server. If the client determines that the packet is intact, it synchronizes its time with the server.

To determine the actual local time, the local clock offset is calculated according to this standard SNTP equation:

$$\text{LocalClockOffset} = ((\text{ReceiveTimestamp} - \text{OriginateTimestamp}) + (\text{TransmitTimestamp} - \text{DestinationTimestamp})) / 2$$

---

**Note** This equation is from RFC 1769. It takes into account network delays and assumes that they are symmetrical.

---

Each of the following variables are specified within the SNTP packet:

- *ReceiveTimestamp* is the local time at which the server receives the latest time request from the client.
- *OriginateTimestamp* is the local time at which the client sends its latest time request.
- *TransmitTimestamp* is the local time at which the server sends the time reply.
- *DestinationTimestamp* is the local time at which the client receives the time reply.

Many factors go into determining whether or not an SNTP packet is valid and whether or not it will be accepted for time synchronization. These factors include:

- Packet size and the fields within the packet that can indicate loss of synchronization (see RFC 1769 for a detailed description of SNTP packet format).
- Verification that the current response is a response to the last request made by the client.
- Verification that the year is greater than or equal to 1995. (This date holds no particular significance.)

If a packet does not meet the preceding qualifications, the packet is disregarded and time is not synchronized.

When the local clock offset has been determined, the following algorithm is used to adjust the time:

- If the local clock time of the client is behind the current time received from the server, W32Time will change the local clock time immediately.
- If the local clock time of the client is more than three minutes ahead of the time on the server, W32Time will change the local clock time immediately.
- If the local clock time of the client is less than three minutes ahead of the time on the server, W32Time will quarter or halve the clock frequency for long enough to bring the clocks into sync. If the client is less than 15 seconds ahead, it will halve the frequency; otherwise, it will quarter the frequency. The amount of time the clock spends running at an unusual frequency depends on the size of the offset that is being corrected.

The Basic Time Synchronization process is illustrated in Figure 1 below.

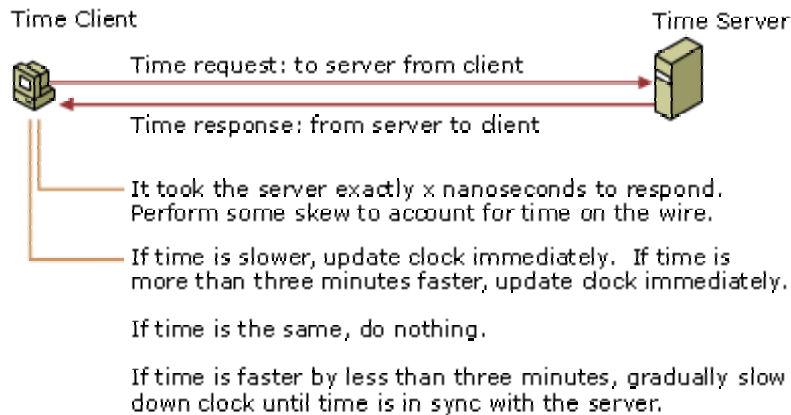


Figure 1 Basic Time Synchronization

W32Time will periodically check its local time with the current time by connecting to the time source, usually the authenticating domain controller. This process starts as soon as the service turns on during system startup. W32Time attempts synchronization every 45 minutes until the clocks have successfully synchronized three times. When the clocks are correctly synchronized, W32Time then synchronizes at eight-hour intervals, unless there is a failure to obtain a timestamp, or a validation failure. If there is a failure, the process starts over from the beginning.

The result is that all computers within the enterprise running W32Time reliably converge to a common time.

## Time Convergence Hierarchy

To synchronize computers on a Windows 2000 network, W32Time must have a reliable time source from which to sync. W32Time can be configured to use one of three methods for synchronization by modifying

entries in the registry (see "W32Time Registry Entries" later in this article). These methods are domain hierarchy based synchronization, using a manually specified synchronization source, or using no synchronization at all. Domain hierarchy based synchronization is the default on every Windows 2000–based computer, and meets the time synchronization requirements specified by the Kerberos V5 protocol. The accuracy of a time server is indicated by a number called the stratum, with the most accurate time server located at stratum one and each level downward in the hierarchy a number one greater than the preceding level. In a Windows 2000 forest, the stratum number of a local computer is determined by how many hops away it is from the external and most accurate time source.

### **Domain Hierarchy Based Synchronization**

Domain hierarchy based synchronization uses the hierarchy of Active Directory, the directory service included with Windows 2000. In domain hierarchy based synchronization, the Flexible Single-Master Operation of the primary domain controller emulator (PDC emulator) is located in the forest root domain and is connected to an external time source. The external time source holds the position of greatest accuracy, or stratum one. The PDC emulator is at stratum two. The forest root domain can also be called the parent domain, and each domain under the parent or forest root can be called a child domain. Any domain controller that accesses time directly from the PDC emulator of the forest root domain is designated as stratum three. As the stratum increases from one, the achievable accuracies degrade depending on the network paths and local clock stabilities. Figure 2 illustrates the hierarchical relationship between computers in a forest.

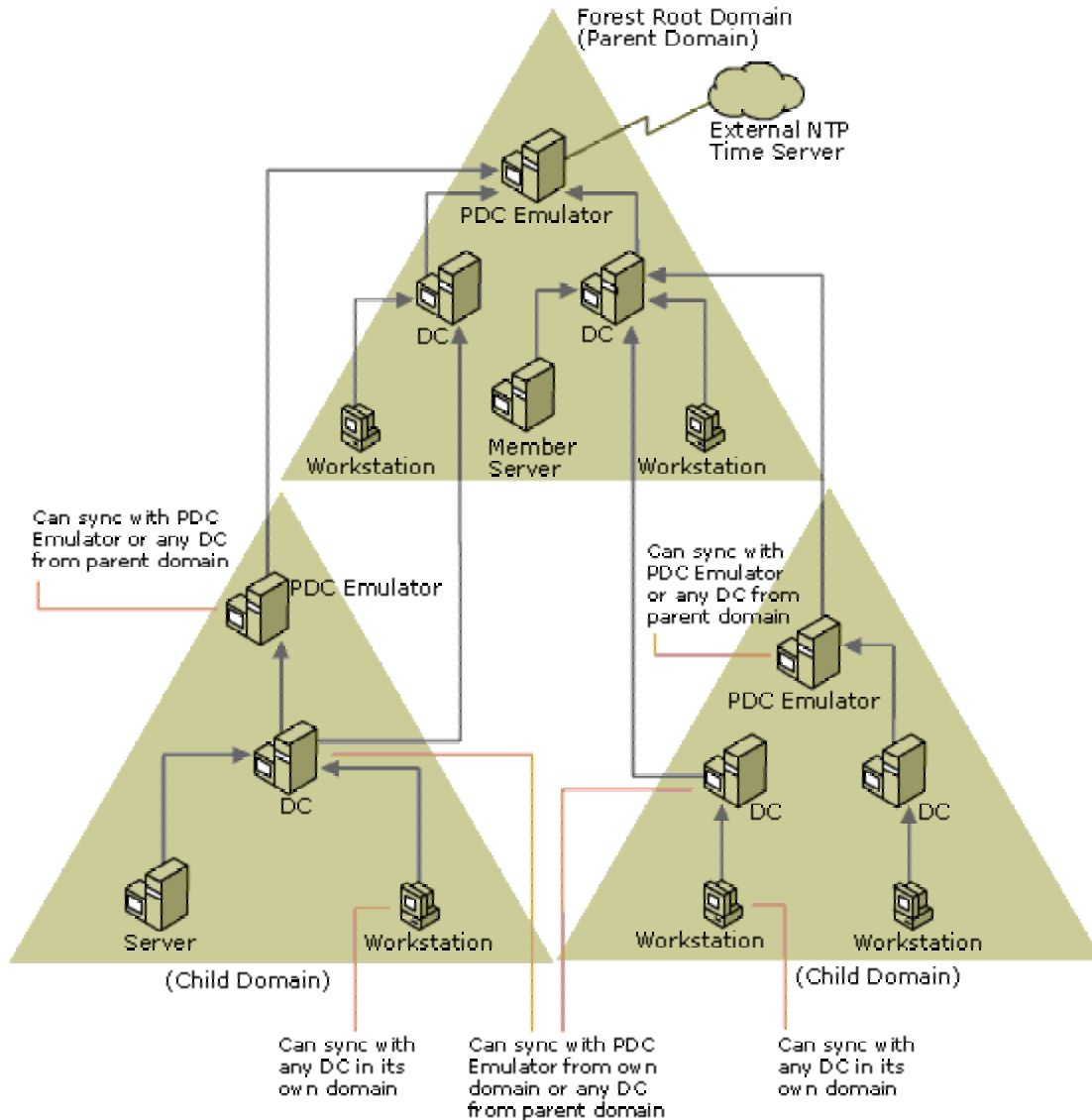


Figure 2 Time synchronization in an Active Directory hierarchy

In this illustration, the PDC emulator located in the parent domain is at stratum 2. It gets its time directly from the external NTP time source, which is located at stratum 1, and therefore keeps the most accurate time in the forest. Domain controllers in the parent domain are at stratum 3. They rely on the PDC emulator for synchronization, and therefore, due to network latency and other factors affecting convergence, have a lesser degree of accuracy. Domain controllers and workstations located at stratum 4 tend to be less accurate than the computers located at stratum 3, and so on. Table 1 identifies computers within a forest and the stratum they usually occupy.

**Table 1 Stratum Hierarchy of Computers in a Forest**

Stratum	Description
1	External NTP time source
2	PDC emulator of the forest root domain
3	Domain controllers in the forest root domain or PDC emulators in child domains
4	Workstations and member servers in the forest root domain or domain controllers in child domains
5	Workstations and member servers in child domains

#### Reliable Time Source Entry

The `ReliableTimeSource` entry in the registry is used to optimize how time is transferred throughout the domain hierarchy. When the reliable time source entry is set to 1 on a domain controller, the Net Logon service announces that computer as a reliable time source when it logs on to the network. When domain controllers look for a time source to synchronize with, they will choose a reliable source if one is available. The primary purpose of this registry entry is to identify a domain controller that is synchronizing with a manually configured time source.

---

**Caution** Setting a computer that is already synchronizing from the domain hierarchy as a reliable time source can create loops in the synchronization tree and cause unpredictable results.

---

Another use of the entry is to identify the root of the time service, which is by default the PDC emulator of the forest root domain. The root of the time service, usually the PDC emulator of the forest root domain, is customarily the computer configured to retrieve time from an external source, and is the authoritative server for the domain. The role of PDC emulator can move between computers, which means that every time the role of PDC emulator moves, the new PDC emulator must be manually configured to point to the external source, and the manual configuration must be removed from the original PDC emulator. To avoid this process, you can set one of the domain controllers in the parent domain as reliable and manually configure just that computer to point to an external source. Then, no matter which computer is the PDC emulator, the root of the time service stays the same and thus remains properly configured. Setting the value of the `ReliableTimeSource` entry to 1 is only useful on a domain controller. (For details about changing the value of `ReliableTimeSource`, see "W32Time Registry Entries" later in this article.)

#### Time Source Identification

A computer uses one of three methods to identify the source against which it synchronizes its time:

1. If the computer is not a member of a domain, it must be configured to use a specific time source. (For more information about configuring a specific time source, see "Configuring W32Time" later in this article.)
2. If the computer is a member server or workstation within a domain, it follows the Active Directory hierarchy and synchronizes its time with any domain controller in its local domain that is currently running W32Time. Each time a computer starts, it will synchronize its time by requesting a time server from Active Directory. Active Directory then returns the DNS name of a single time server.

The client resolves the DNS name to the IP address of the time server and synchronizes its time with that of the time server.

3. If the computer is a domain controller, it makes up to three queries to locate another domain controller with which to synchronize. The queries are designed to identify a time source with certain attributes. These attributes, in order of most to least important, are:
  - Whether a domain controller is in the same site.
  - Whether the domain controller is marked as a reliable time source.
  - Whether the domain controller is in the parent domain.
  - Whether the domain controller is a PDC emulator.

Computers choose a time source according to a specific order of preference. This order, from most preferred to least preferred, is listed in Table 2.

**Table 2 Preferences for Selecting a Time Source**

Preference	Computer	Location	Reliability of Time Source
1	Parent domain controller	In-site	Reliable
2	Local domain controller	In-site	Reliable
3	Parent domain controller	In-site	Not reliable
4	Local PDC emulator	In-site	Not reliable
5	Parent domain controller	Out-of-site	Reliable
6	Local domain controller	Out-of-site	Reliable
7	Parent domain controller	Out-of-site	Not reliable
8	Local PDC emulator	Out-of-site	Not reliable

---

**Note** If priority 6 (local domain controller, out-of-site, reliable) is found, the existence of priority 4 (local PDC emulator, in-site, not reliable) will be ignored. Also, a computer will never choose itself to sync from, so if this computer is the local PDC emulator, it will not choose priority 4 or priority 8.

---

If a computer is the PDC emulator in the forest root domain, there is no computer in the domain hierarchy from which it can synchronize. The time on the PDC emulator will be governed by the computer's internal hardware clock, unless an external time source has been manually configured using the Net Time command. (For information about how to manually configure an external time source, see "Configuring W32 Time" later in this article.)

### Manually Specified Synchronization

Manually specified synchronization allows you to configure a client to request time from a specific time source by using the **net time** command: `/setsntp:ntpserver`. (For more information about Net Time configuration, see "Net Time" later in this article.) This manual configuration requires extensive administration to implement over a large network and might compromise network security. Manually specified time sources are not authenticated, and therefore can enable an attacker to manipulate the time source and then start Kerberos V5 replay attacks. Also, if a computer does not synchronize with its domain controller, the two might be out of sync. This causes Kerberos V5 authentication to fail, which in turn causes other actions requiring network authentication, such as printing or file sharing, to fail. When only one

computer in the forest root domain is getting time from an external source, all computers within the forest remain synchronized to each other, making replay attacks difficult.

### **No Specified Synchronization**

It is possible to use third-party software to synchronize with an external source outside of a forest, and still use W32Time to securely distribute time within the forest. You can do this by disabling synchronization while leaving W32Time active on the server. To do this, stop W32Time and install the third-party software on the PDC emulator in the forest root. Then disable the third-party software's ability to serve time by setting the value for the **Type** entry to "NoSync" in the registry, and then restart W32Time. (For details about changing registry entries, see "W32Time Registry Entries" later in this article.) The domain will be securely synchronized to the PDC emulator's clock, while the PDC emulator's clock will be using the third-party software to synchronize with an accurate source outside of the forest.

---

## W32Time Service Tools

Two W32Time service tools located in the \system32 folder can be accessed from the command line in Windows 2000. The Net Time tool allows you to configure the W32Time service. The W32tm tool is used primarily for diagnostic purposes and does not allow you to make any configuration changes.

---

**Note** It is important to remember that if you make any configuration changes to the W32Time service using any of the tools described in this section you must stop and restart the service for the changes to take effect. By default, only administrators have the permission to stop W32Time.

---

### Net Time

You can use Net Time to manually configure the time source for a computer. Although W32Time starts automatically at system startup, you can manually stop and start the service by typing the appropriate syntax at a command prompt. You can also use Net Time to check the time on a remote computer and set the clock of the local computer to match the remote computer.

#### To manually stop W32Time

At the command prompt, type:

```
net stop w32time
```

#### To manually start W32Time

At the command prompt, type:

```
net start w32time
```

Typing **net time /?** at a command prompt provides the following syntax:

```
Net time      [\\computername | /DOMAIN:domainname]
              /RTSDOMAIN[:domainname] [/set]
              \\computername /querysnTP
              \\computername /setsntp[:ntp server list]
```

---

**Note** It is recommended that only one IP address or DNS server name be used at a time with this command. W32Time will only use the first IP address or DNS server name specified even if more than one listed.

---

Table 3 lists the parameters available for Net Time.

**Table 3 Net Time Parameters**

Parameter	Description
net time	Displays the time on a time server in this computer's domain.
net time \\ <b>computername</b>	Displays the time on <i>computername</i> .
net time /domain: <b>domainname</b>	Displays the time on a domain controller in <i>domainname</i> .
net time /domain /set	Sets this computer's time to match the time on a domain controller in this computer's domain.
net time /rtsdomain: <b>domainname</b>	Displays the time on a time server in <i>domainname</i> . Note that the

	/rtsdomain flag does <i>not</i> actually require a time server to be marked as reliable.
net time /querysnTP	Displays the manually configured time source for this computer, if there is one.
net time /setsntp: <i>ntpserver</i>	Sets the manually configured time source for this computer. It is important to note that only one DNS name or IP address can be specified.
net time /setsntp	Clears the manually configured time source for this computer, if one has been specified, and then determines the time source from the domain hierarchy.

## W32tm

The W32tm tool is used for diagnosing problems that can occur with W32Time. If you are going to use the W32tm tool on a domain controller, it is necessary to stop W32Time. Running W32tm and W32Time at the same time on a domain controller generates an error because both are attempting to use the same UDP port. When you finish using W32tm, W32Time must be restarted.

Typing **w32tm /?** at a command prompt provides the following syntax:

**W32tm [-tz | -s *[computer]* | -adj | -adjoff | -source | -once] [-test]  
[-v] [-p *port*] [-period *freq*]**

Table 4 describes the functions of the W32tm tool.

**Table 4 W32tm Primary Parameters**

Parameter	Description
-tz	Prints the local time zone information and exit.
-s <i>computer</i>	Forces <i>computer</i> (or the local computer if none is specified) to resynchronize, then exits.
-adj	Sets the computer's system clock frequency to the last frequency determined during synchronization, then exits.
-adjoff	Sets the computer's system clock frequency as the system default, then exits.
-source	Chooses a synchronization source, then exits. Note that you choose a source before each synchronization, so this is useful only in showing that a source cannot be found.
-once	Synchronizes once, and then exits. Otherwise, runs continuously as a client, synchronizing the local clock until ctrl-c is pressed.

The program will also run as a server any time the service would (for example, when it is on a domain controller, or when the data string for the value of the LocalNTP registry entry is changed from 0 to 1).

Table 5 shows optional parameters that can be used in conjunction with the parameters shown in Table 4.

**Table 5 W32tm Optional Parameters**

Parameter	Description
-test	Prevents the time on the local system from being modified.
-v	Prints out a detailed description of what the program is doing. This is usually needed

	because otherwise the program produces no output. The exceptions are -s and -tz.
-p <i>port</i>	Sets the server port.
-period <i>freq</i>	<p>Sets the sync period just as in the registry. (For more information, see “W32Time Registry Entries” later in this article). That is:</p> <p>0 = once per day  65535 = once every two days  65534 = once every three days  65533 = once every week (seven days)  65532 = once every 45 minutes until you get three good syncs, then once every eight hours (three/day)  65531 = once every 45 minutes until you get one good sync, then once every day</p> <p>Otherwise <i>freq</i> specifies the number of times per day. If you choose to add a value other than any of those specified above, you must use this option.</p>

Examples:

- `w32tm -s computer` - Makes *computer* resynchronize. The computer resynchronizes with the computer with which it was set to sync at system startup. Note that a return value of zero (0) means that communication with *computer* was successful and *computer* made an attempt to resync. It does not mean that the resync attempt was successful.
- `w32tm -once -test -v` - Watches an attempt to sync this computer to its time source, but does not actually change the time. This shows where in the process the time service client is failing. Note that when the time source is chosen automatically, a different source might be chosen each time.

## Configuring W32Time

Achieving optimal performance from your network implementation of W32Time requires some fine-tuning. Net Time and a registry editor are both valuable tools for configuring W32Time. You can configure W32Time to designate an external time source or to start or stop manually rather than automatically at system startup. In addition, you can adjust the setting for the Kerberos default time skew, which defines the range of time allowed for a server to accept authenticators from a particular client.

### W32Time Registry Entries

All registry entries for the W32Time service are in the HKEY LOCAL MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Parameters subkey. Table 6 gives a complete list of all configurable registry entries associated with W32Time. In addition to the Parameters subkey, there are two additional subkeys located under HKEY LOCAL MACHINE\SYSTEM\CurrentControlSet\Services\W32Time. These are Enum and Security. It is important to note that these subkeys are service specific and are used only by the Service Control Manager. These subkeys must never be configured manually.

**Note** Remember, after changing the configuration, you must stop and restart the W32Time service for the changes to take effect.

**Caution** Do not edit the registry directly unless you have no alternative. The registry editors bypass standard safeguards, allowing settings that can degrade performance, damage your system, or even require you to reinstall Windows. You can safely alter most registry settings using the programs in Control Panel or Microsoft Management Console (MMC). If you must edit the registry directly, back it up first. Read Registry Editor Help for more information.

**Table 6 W32Time Registry Entries**

Entry Name	Data Type	Description and Values
ReliableTimeSource	REG_DWORD optional	Used to indicate that this computer has reliable time. 0 = Do not mark computer as reliable. [default] 1 = Mark computer as reliable. This is only useful on a domain controller.
Period	REG_DWORD or REG_SZ	Used to control how often the time service synchronizes. If a value is given, it must be one of the special values listed below. 65531, "DailySpecialSkew" = once every 45 minutes until successful one time, then once every day 65532, "SpecialSkew" = once every 45 minutes until successful three times, then once every eight hours (three times per day) [default] 65533, "Weekly" = once every week (seven days) 65534, "Tridaily" = once every three days 65535, "BiDaily" = once every two days 0 = once per day <i>freq</i> = <i>freq</i> times per day. If you choose to add a value other than any of those specified above, you must use this option.

AvoidTimeSyncOnWan	REG_DWORD optional	Used to prevent the computer from syncing from a computer that is in another site and thus connected by a costly temporary connection. 0 = The site of the time source is ignored. [default] 1 = The computer will not synchronize with a time source that is in a different site.
LocalNTP	REG_DWORD	Used to start the SNTP server. 0 = Do not start server unless this computer is a domain controller. [default] 1 = Always start server.
Type	REG_SZ	Used to control how a machine synchronizes. Nt5DS = Synchronize to domain hierarchy or manually configured source. [default] NTP = Synchronize to manually configured source. NoSync = Do not synchronize.
NtpServer	REG_SZ optional	Used to manually configure the time source. This can be set to the DNS name or IP address of the server from which to sync. Only one DNS name or IP address can be specified. This can be modified from the command line. See <a href="#">net time</a> . [default = blank]
GetDcBackoffMinutes	REG_DWORD optional	The initial number of minutes to wait before looking for a domain controller (time source) if the last attempt to find a domain controller failed. [default = 15]
GetDcBackoffMaxTimes	REG_DWORD optional	The maximum number of times to double the backoff interval when successive attempts to find a domain controller fail. An event is logged every time a wait of the maximum length occurs. If the value of this entry is 0, then the wait between successive attempts is always the minimum and no event is logged. [default = 7]  The time service tries to find a domain controller according to its usual sync schedule, but if the backoff interval has not expired, then that attempt will be skipped. For example, if given the default values, the backoff interval will follow this pattern: 15 minutes, 30 minutes, 1 hour, 2 hours, 4 hours, 8 hours, 16 hours, 16 hour, 16 hours, etc. However, the time service will only attempt to sync on 45 minute intervals, so the attempts to find a domain controller will actually occur after 45min, 1 hour 30 minutes, 2 hours 15 minutes, 4 hours 30 minutes, 8 hours 15 minutes, 16 hours 30 minutes, etc.

The **Adj** and **msSkewPerDay** entries are used to preserve information about the computer's clock between system start-ups, and they must never be manually edited.

## Designating an External Time Source

The Net Time tool allows you to designate an external time source. It is important to note that even though the net time /? command returns a syntax that specifies that an "NTP List" can be designated, it is highly

recommended that you only list one DNS name or IP address at a time. W32Time only recognizes the first DNS name or IP address listed and listing more than one might return an error.

### To designate an external time source

At the command prompt, type:

```
net time /setsntp:DNSName
```

– or –

```
net time /setsntp:IPAddress
```

## Manually Starting and Stopping W32Time

By default, W32Time starts automatically at system startup. However, you can start or stop W32Time manually by accessing services in Administrative Tools or by using the Net Time tool.

### To manually start W32Time using the graphical interface

1. From the Start menu, point to Settings, and then click Control Panel.
2. Double-click Administrative Tools, and then double-click Services.
3. Select Windows Time from the list of services.
4. On the Action menu, click Start to begin the service.

### To manually stop W32Time using the graphical interface

1. Follow steps 1 through 3 in the previous procedure.
2. On the Action menu, click Stop to discontinue the service.

### To manually start W32Time using Net Time

- At the command prompt, type:  

```
net start w32time
```

### To manually stop W32Time using Net Time

- At the command prompt, type:  

```
net stop w32time
```

## Adjusting the Kerberos Time Skew Variable

In Kerberos, the variable of "time skew" defines how "loosely" two participants' clocks need to be synchronized. By default, this variable is set at five minutes. However, if you wanted more protection from replay attacks and your network connectivity allowed for it, you could shorten the time skew variable. On the other hand, if your network connectivity was poor and five minutes was not enough time for the clients on your network to be authenticated, you may want to lengthen the variable.

### To change the Kerberos time skew variable

1. From the Start menu, point to Settings, click Control Panel, double-click Administrative Tools, and then double-click Domain Security Policy.
2. Expand Security Settings, Account Policies, and Kerberos Policy.
3. Right-click Maximum tolerance for computer clock synchronization.
4. Click Security.
5. In the Security Policy dialog box, change the maximum tolerance variable.

---

## Troubleshooting Time-Related Errors

You might encounter several types of errors while working with W32Time. Recognizing these errors and knowing what steps to take to fix them will be helpful to you as you administer W32Time on your network.

When W32Time is running on the PDC emulator, it commonly sends messages to the Event Log indicating that it has no time source. When this error occurs, the solution is to either configure a manual time source for the PDC emulator of the forest root domain or set the PDC emulator to not synchronize at all by changing the value for the **Type** entry in the registry to "NoSync".

Many other errors affecting W32Time are specific to the Net Time tool. Some of these errors are described below.

### Unable to Locate Time Server

When using the Net Time tool, you might receive the following error message:

Could not locate a time server.  
More help is available by typing NET HELPMSG 3912.

This error message can occur even if you set a valid Simple Network Time Protocol (SNTP) time server using the `net time /setsntp` command, and a network connection to the external time server exists. The manually specified time source might not be in the local workgroup or in the domain, or it might not be announcing itself as a time server. Receiving this message does not mean that the time service is not synchronizing time.

To verify that the time service is synchronizing time

- At the command prompt, type:  
`w32tm -v -once -test`

This error message can also occur when the time service was not stopped before the configuration change was made. To avoid this error, you must stop the W32Time service before using the **net time /setsntp** command, and then restart the service when the change has been made.

This error can also be generated when UDP port 123 is closed on the firewall or router between the client and the server or is being used by another service. UDP port 123 is the default SNTP port. Note that the other service using UDP port 123 might be W32Time, in which case stopping and restarting the W32Time service probably fixes the problem.

### Access Denied

While running `net time \\computer /set` you receive an access denied error.

This error occurs when a remote procedure call (RPC) fails to authenticate, usually because a user does not have permission to access the remote computer and run Net Time. If you know the user name and password of an account that does have access rights, you can establish credentials by using the **net use** command.

#### To establish credentials using net use

- **At the command prompt, type:**  
`net use \\computer\ipc$ \user: username. *`

## Failure to Bind

When using the net time command, you receive the following error message:

Bind Failed: 0x80072740

Only one usage of each socket address *protocol/network address/port* is normally permitted.

Two instances of the same service are trying to start using the same port. The W32Time service is already using UDP port 123 (the default port for the time service); therefore the W32tm tool is not able to use the port. You must stop W32Time by using net stop before running the W32tm tool.

## Unable to Log On to the Network

When you attempt to log on to the network you might receive the following error message:

The system cannot log you on due to the following error:

There is a time difference between the Client and Server.

Please try again or consult your system administrator.

This behavior can occur if the time or date is not synchronized between your computer and the domain to which you are attempting to log on. If the time or date on a client is not synchronized with the authenticating domain controller, Kerberos validation does not succeed.

Because Kerberos is the only form of logon authentication between two Windows 2000-based computers, the logon attempt does not succeed.

To resolve this issue, log on to your computer locally by using an account with administrative privileges and set the time and date to match the time and date on the domain controller that validates your logon process.

---

## FAQ

### Why has UTC been designated the official acronym for Coordinated Universal Time?

Before 1970, *Greenwich Mean Time* was the correct term for time with no time zone offset. In 1970, the International Telecommunication Union (ITU), comprised of a group of international technical experts, devised the coordinated universal time (UTC) system. The ITU wanted a single abbreviation for Coordinated Universal Time in order to minimize confusion caused by language barriers. Because members of the ITU could not come to a unanimous agreement on either the English acronym (CUT) or the French acronym (TUC), the acronym UTC became the agreed-upon compromise.

### What external NTP time servers are available for synchronization?

Many methods exist for synchronizing a computer's clock. External time servers allow users to synchronize computer clocks by means of dial-up, network, and radio links.

NIST, the National Institute of Standards and Technology, located in Boulder, Colorado, provides the Automated Computer Time Service (ACTS), which can set a computer clock with an uncertainty of less than 10 milliseconds. The U.S. Naval Observatory (USNO) Time Service Department in Washington D.C. is another source for accurate time synchronization in the United States. Many other sites exist throughout the world that can be used for time synchronization. To find them, run a search for "time synchronization" on the Internet.

### Is it necessary to synchronize time across forests?

Currently, no time protocols in Windows 2000 work across forests and require that forests be in sync. However, PDC emulators in separate, independent forests need to be synchronized with the same globally correct time in order to provide for accurate time stamping on e-mail, log files, etc.

### Can a time server be run on any computer?

You can designate any computer as a time server by changing the value of the LocalNTP entry in the registry from 0 to 1.

All registry entries for the Windows Time Service are in the HKEY LOCAL MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Parameters subkey. See Table 6 earlier in this article for a complete list of all registry entries associated with W32Time.

It is important to note that the automatic discovery mechanism in the time service client *never* chooses a computer that is not a domain controller. Clients must be manually configured to use any server that is not a domain controller.

---

**Caution** Do not edit the registry directly unless you have no alternative. The registry editors bypass standard safeguards, allowing settings that can degrade performance, damage your system, or even require you to reinstall Windows. You can safely alter most registry settings using the programs in Control Panel or Microsoft Management Console (MMC). If you must edit the registry directly, back it up first. Read Registry Editor Help for more information.

---

### I know my computer's time is incorrect. Why hasn't it synchronized yet?

A clock on a computer does not adjust itself immediately. A computer synchronizes with the time server at system startup and then every 45 minutes until it has successfully synchronized three times. When the clocks are correctly synchronized, they sync again after eight hours, and every

eight hours after that. (For a description of how to manually change this setting, see "W32Time Registry Entries" earlier in this article.)

---

## Appendix A: Microsoft Windows IP Security

Microsoft Windows IP Security uses industry-standard encryption algorithms and a comprehensive security management approach to provide security for all TCP/IP communications on both sides of an organization's firewall. The result is a Windows 2000 Server end-to-end security strategy that defends against both external and internal attacks. IPsec exists below the transport layer, making it transparent to applications and users, which means that there is no need to change network applications on a user's desktop when IPsec is implemented in the firewall or router.

IPsec is a suite of protocols that allow secure, encrypted communication between two computers over an insecure network. The encryption is applied at the IP network layer, which means that it is transparent to applications (in this case, time-related applications) that use specific protocols for network communication. IPsec provides end-to-end security, meaning that the IP packets are encrypted by the sending computer, are unreadable en route, and can be decrypted only by the recipient computer. Because of a special algorithm for generating the same shared encryption key at both ends of the connection, the key does not need to be passed over the network. With IP Security, only the sender and recipient know the security key. If the authentication data is valid, the recipient knows that the communication came from the sender and that it was not changed in transit.

---

**Note:** IPsec can be implemented on a firewall as well as on the time source device or machine.

---

### Creating IPsec Policies

You can use the IP Security Policy administrative snap-in to create, modify, and activate IPsec policies. You can create a console by adding the IP Security Policies on Local Machine snap-in to a blank console or by double clicking the Local Security Policy icon located in Administrative Tools. You can also access the IP Security Policies snap-in through the Group Policy administrative console by navigating to **Computer Configuration\Windows Settings\Security Settings\IP Security Policies**. Within the Group Policy administrative console, you can add the IP Security Policy extension snap-in, which can be used to troubleshoot policy precedence issues and determine the exact set of policies that IPsec clients are using.

IPsec can utilize the following three authentication methods:

- Kerberos
- Pre-shared Key
- Certificates

IPsec can only use Certificates or Pre-shared Key to encrypt sessions to external time sources. IPsec cannot use Kerberos to encrypt sessions to external time sources because Kerberos itself relies on time for authentication, thus a catch22 is created if time becomes skewed.

The time source must also support IPsec as well as the authentication method (Pre-shared Key or Certificates) used by the client. IPsec interoperability must be verified with the time source to ensure that it handles the IKE negotiation parameters and the rekey process properly (for example when the IPsec SAs idle out).

Microsoft has created the following IPsec policies. These policies can be used to protect your configuration if both your time client and your external time source are running the Windows 2000 operating system.

However, in most real world configurations, the external time source is not running Windows 2000. Many external time sources are either Unix based or are configured using a special dedicated device. To implement policy you will have to find the equivalent IPSec configuration to be used on the external time source.

If you do not have the equivalent IPSec configuration for your external time source, you can use these policies as a learning device to understand how IPSec works. After the policies have been established on both a Windows 2000 client and server, type the following at a command prompt:

- `C:\w32tm /resync`

This command causes the time service to synchronize immediately. If you monitor your network traffic, you will notice that the ISAKMP traffic has been encrypted. If you remove the policies and perform the same actions, you will notice that the traffic is no longer encrypted.

You can install the policies on the local machine by importing them into the IP Security Policies on Local Machine snap-in. You can also import them into the Group Policy snap-in so that IPSec can be deployed onto many computers at once.

Follow the directions below to import the IPSec policies provided.

### **To import IPSec Policies to your local computer**

1. Click Start, click Run, type **mmc**, and click **OK**.
2. On the console menu, click **Add/Remove Snap-in**.
3. In the Add/Remove Snap-in dialog box, click **Add**.
4. Select the IP Security Policy Management snap-in, and click **Add**.
5. Select Local Computer, and click **Finish**.
6. Click **Close**, and click **OK**.
7. In the left pane, right-click **IP Security Policies on Local Machine**, click **All Tasks**, and select **Import Policies**.

Browse to the location where you have stored the policies on your computer, select the policy that you want to import, and click **Open**.

8. The policy appears as a separate line item in the right pane.

The policies provided here are executable on any Windows 2000 computer (Professional or Server).

The IPSec policies provided include the following parameters:

- Time client: Windows 2000 Professional.
- Time Source: Windows 2000 server listening on port 123.
- Authentication method: Pre-Shared Keys.
- Encryption: 3DES/SHA1.
- No tunneling used.

### **Time Client Policy**

Import the following IPsec policy on the time client to block all incoming traffic from the Time Source except UDP packets destined for port 123. This policy enforces 3DES/SHA1 encryption.



SNTPClient.ipsec

### **Time Source Policy**

Import the following IPsec policy on the time source to require that information from a time client be secure. This policy only allows packets from port 123.



SNTPSource.ipsec

For more information about and step-by-step instructions for creating IP Security policies by using the IP Security Policies snap-in for policy configuration, see "Step-by-Step Guide to Internet Protocol Security (IPsec)" at <http://www.microsoft.com/windows2000/techinfo/planning/security/ipsecsteps.asp>.

You can also find information about several Unix implementations of IPsec at the following links:

- Japanese KAME' implementation for freeBSD, openBSD, netBSD  
<http://www.kame.net/>
- Linux  
<http://www.freeswan.org/>
- OpenBSD  
<http://www.openbsd.org/faq/faq13.html#13.10>
- Solaris  
<http://www.sun.com/software/white-papers/wp-s8security/#2.1>
- HP-UX  
<http://www.hp.com/security/products/ipsec9000/papers/datasheet/>

---

## Related Links

For the latest information about Windows 2000 Server, see the [Windows 2000 Server Web site](http://www.microsoft.com/windows2000/server) at <http://www.microsoft.com/windows2000/server>.