

How the Active Directory Replication Model Works

In this section

- [Active Directory Replication Model Architecture](#)
- [Active Directory Replication Model Physical Structure](#)
- [Active Directory Data Updates](#)
- [Domain Controller Notification of Changes](#)
- [Identifying and Locating Replication Partners](#)
- [Urgent Replication](#)
- [Network Ports Used by Active Directory Replication](#)
- [Related Information](#)

Active Directory data takes the form of objects that have properties, or attributes. Each object is an instance of an object class, and object classes and their respective attributes are defined in the Active Directory schema. The values of the attributes define the object, and a change to a value of an attribute must be transferred from the domain controller on which it occurs to every other domain controller that stores a replica of that object.

Thus, Active Directory replicates directory data updates at the attribute level. In addition, updates from the same directory partition are replicated as a unit to the corresponding replica on the destination domain controller over the same connection to optimize network usage.

The information in this section applies to organizations that are designing, deploying, or operating an Active Directory infrastructure that satisfies the following requirements:

- A Domain Name System (DNS) infrastructure is in place that manages the name resolution for domain controllers in the forest. Active Directory-integrated DNS is assumed, wherein DNS zone data is stored in Active Directory and is replicated to all domain controllers that are DNS servers.
- All Active Directory sites have local area network (LAN) connectivity.
- IP connectivity is available between all datacenter locations and branch sites.

The limits for data that can be replicated in one replication cycle are as follows:

- Values that can be transferred in one replication cycle (replication of the current set of updates between a source and destination domain controller): no limit.
- Values that can be transferred in one replication packet: approximately 100. Replication exchanges continue during the course of one replication cycle until no values are left to send.
- Values that can be written in a single transaction: 5,000. The effect of this limit depends on the forest functional level:
 - Windows 2000 forest functional level: The minimum unit of replication at this level is the entire attribute. Therefore, changes to any value in the linked, multivalued **member** attribute results in replicating the entire attribute. For this reason, the supported size of group membership is limited to 5,000.
 - Windows Server 2003 or Windows Server 2003 interim forest functional level: The minimum unit of replication is a single value of a linked, multivalued attribute. Therefore, the limitation on group membership is effectively removed.

This section covers the interactions that take place between individual domain controllers to synchronize directory data in an Active Directory forest.

[Back to Top](#)

Active Directory Replication Model Architecture

Active Directory replication operates within the directory service component of the security subsystem. The directory service component, Ntdsa.dll, is accessed through the Lightweight Directory Access Protocol (LDAP) network protocol and LDAP C application programming interface (API) for directory service updates, as implemented in Wldap32.dll. The updates are transported over Internet Protocol (IP) as packaged by the replication remote procedure call (RPC) protocol. Simple Mail Transfer Protocol (SMTP) can also be used to prepare non-domain updates for Transmission Control Protocol (TCP) transport over IP.

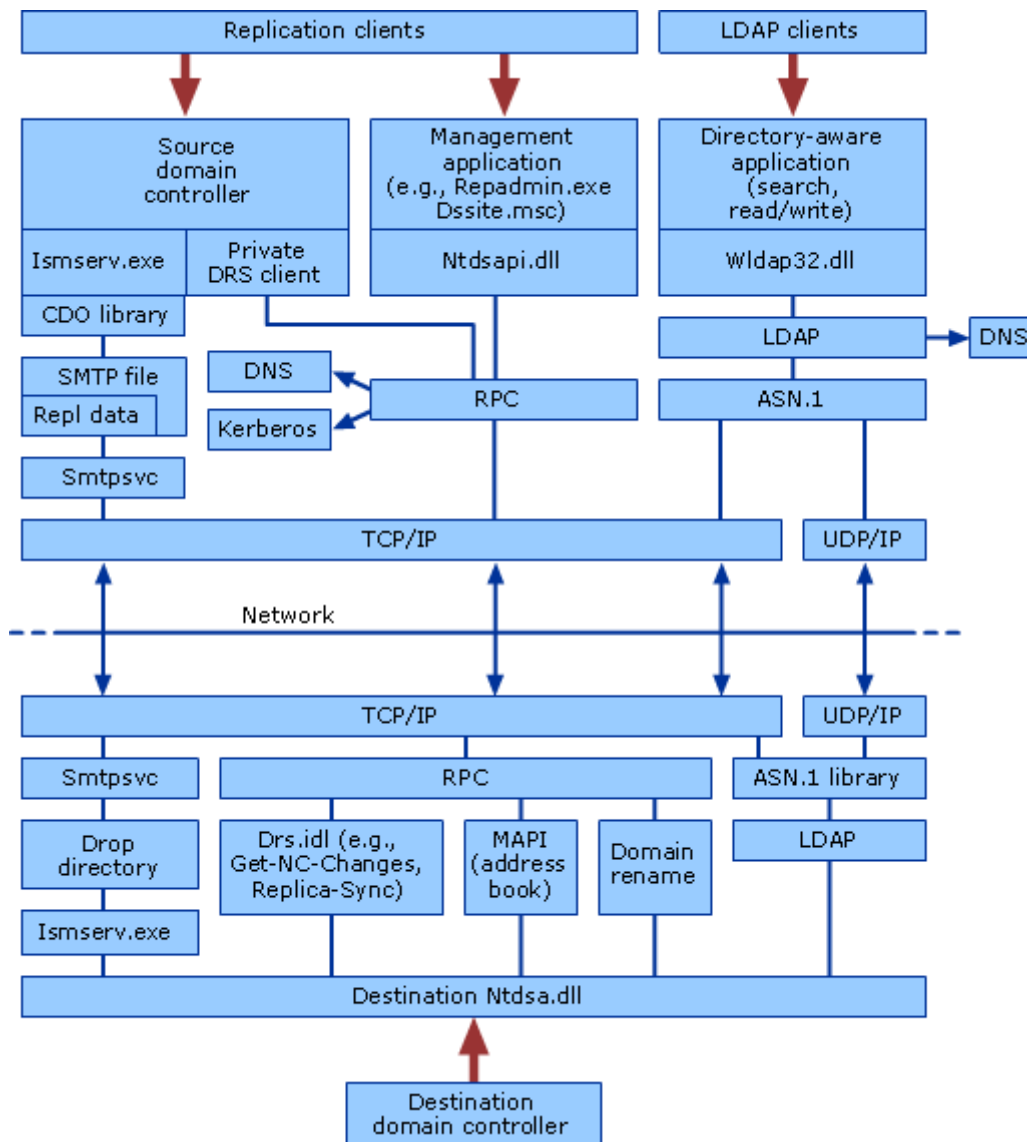
The Directory Replication System (DRS) client and server components interact to transfer and apply Active Directory updates between domain controllers.

When SMTP is used for the replication transport, Ismserv.exe on the source domain controller uses the Collaborative Data Object (CDO) library to build an SMTP file on disk with the replication data as the attached mail message. The message file is placed in a queue directory. When the mail is scheduled for transfer by the mail server application, the SMTP service (Smtpsvc) delivers the mail message to the destination domain

controller over TCP/IP and places the file in the drop directory on the destination domain controller. Ismserv.exe applies the updates on the destination.

The following diagram shows the client-server architecture for replication clients and LDAP clients.

Replication and LDAP Client-Server Architecture



The following table describes the replication architecture components.

Replication Architecture Components

Component	Description
Ntdsapi.dll	Manages communication with the directory service over RPC.
Private DRS client	Private version of Ntdsapi.dll that runs on domain controllers to make RPC calls for replication.
Wldap32.dll	Client library with APIs for access to directory service.
Asn.1	Encodes and decodes LDAP requests for transport over TCP/IP or UDP/IP.
Drs.idl	Set of functions for replication (for example, Get-Changes) and maintenance (for example, Get replication status)
MAPI (address book)	Entry protocol for address book applications such as Microsoft Outlook.
Domain rename	Carries out domain rename instructions.
Ntdsa.dll	The directory service module, which supports the Windows Server 2003 and Windows 2000 replication protocol and LDAP, and manages partitions of data.
ISMServ.exe	Prepares replication data in e-mail message format for SMTP protocol transport.
CDO library	Used by Ismserv.exe to package replication data into a mail message.

Smtpsvc SMTP service.

Note

- If Windows NT 4.0 backup domain controllers (BDCs) are operating in the forest, Windows NT4 Net APIs provide an entry to the security accounts manager (SAM) on the primary domain controller (PDC) emulator.

The protocols that are used by Active Directory replication are described in the following table. RPC and SMTP are the replication transport protocols. LDAP is a directory access protocol, and IP is a network wire protocol.

Active Directory Access and Replication Protocols

Protocol	Description
LDAP	The primary directory access protocol for Active Directory. Windows Server 2003 family, Windows XP, Windows 2000 Server family, and Windows 2000 Professional clients, as well as Windows 98, Windows 95, and Windows NT 4.0 clients that have the Active Directory client components installed, use LDAP v3 to connect to Active Directory.
IP	Routable protocol that is responsible for the addressing, routing, and fragmenting of packets by the sending node. IP is required for Active Directory replication.
Replication RPC	The Directory Replication Service (Drsuapi) RPC protocol, used in the enabling of administration and monitoring of Active Directory replication, to communicate replication status and topology and network topology from a client running administrative tools to a domain controller. RPC is required by Active Directory replication.
Replication Simple Mail Transfer Protocol (SMTP)	Replication protocol that can be used by Active Directory replication over IP network transport for message-based replication between sites only and for non-domain replication only.

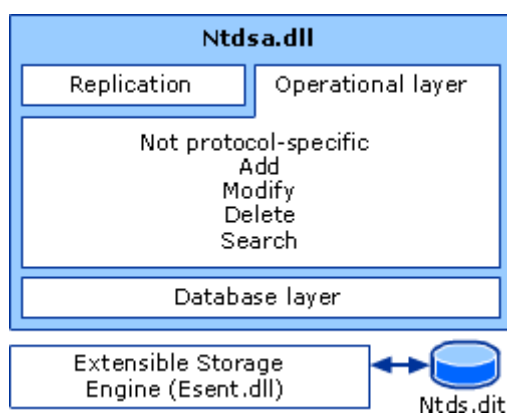
Replication Subsystem

Within the directory service component of the Active Directory architecture, the replication subsystem interacts with the operational layer to implement replication changes on the destination domain controller. The replication subsystem also determines the changes that a replication partner already has or those that are needed.

The database layer manages the database capability of the directory service. The extensible storage engine (Esent.dll) communicates directly with individual records in the directory data store.

The following diagram shows the replication subsystem components.

Replication Subsystem Components



The components of the replication subsystem are described in the following table.

Replication Subsystem Components

Component	Description
Ntdsa.dll	Directory system agent (DSA), which provides the interfaces through which directory clients and other directory servers gain access to the directory database.
Replication	Directory Replication System (DRS) interface, which communicates with the database through RPC.
Operational layer	Performs low-level operations on the database without regard for protocol.

Database layer	API that resides within Ntdsa.dll and provides an interface between applications and the directory database to protect the database from direct interaction with applications.
Extensible storage engine (ESE)	Manages the tables of records, each with one or more columns that comprise the directory database.
Ntds.dit	The directory database file.

[Back to Top](#)

Active Directory Replication Model Physical Structure

The Active Directory replication model components that determine how Active Directory replication functions between domain controllers are associated with mechanisms that effect automatic transfer of changes between replicating domain controllers, as described in the following table.

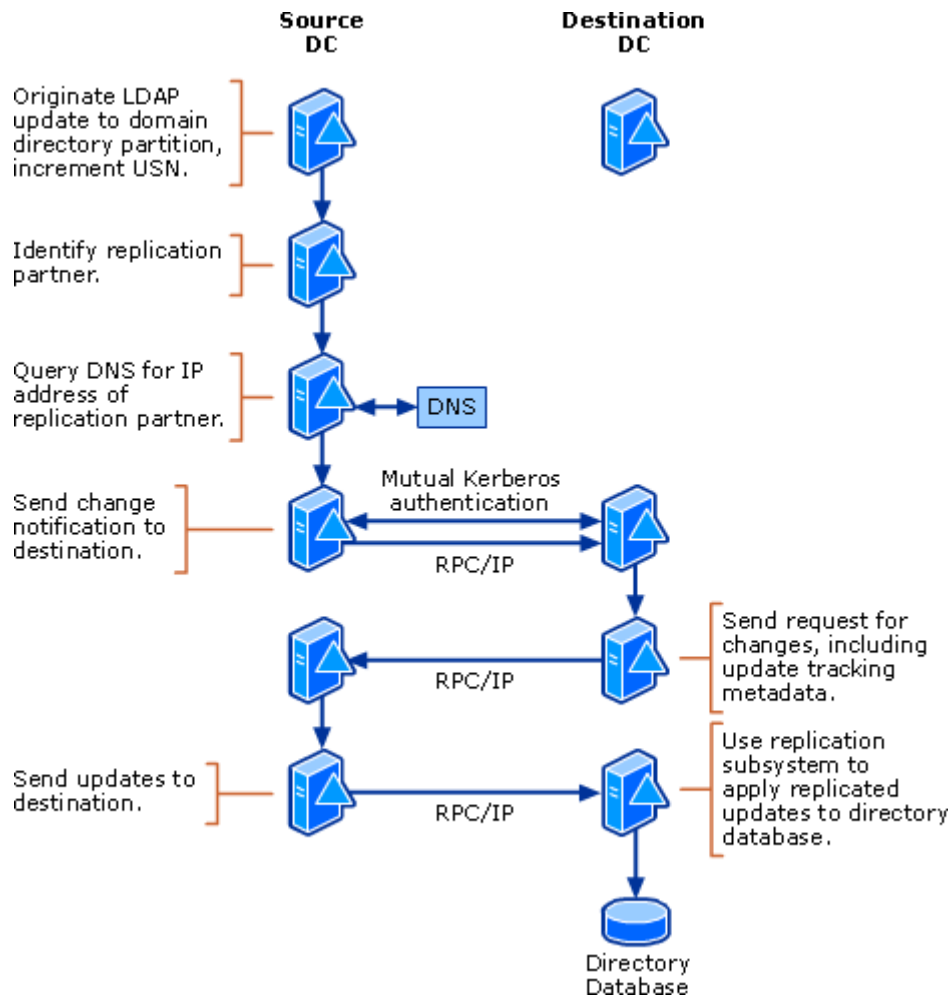
Active Directory Replication Model Components and Related Mechanisms

Component	Description	Related Mechanisms
Multimaster replication	All domain controllers accept LDAP requests for changes to attributes of Active Directory objects for which they are authoritative, subject to security constraints that are in place. Each originating update is replicated to one or more other domain controllers, which record it as a replicated update.	LDAP update Directory partitions Change notification Change tracking Conflict resolution
Pull replication	When an update occurs on a domain controller, it notifies its replication partner. The partner domain controller responds by requesting (pulling) the changes from the source domain controller.	DNS name resolution Kerberos authentication Change tracking Change notification Change request
Store-and-forward replication	Domain controllers store changes received from replication partners and forward the changes to other domain controllers so that the originating domain controller for each change is not required to transfer changes to every other domain controller that requires the change.	Change tracking Kerberos authentication DNS name resolution Change notification Change request
State-based replication	Active Directory replication is driven by the difference between the current "state" (the current values of all attributes) of the directory partition replica on the source and destination domain controllers. This state includes metadata that is used to resolve conflicts and to avoid sending the full replica on each replication cycle.	Change-tracking metadata: <ul style="list-style-type: none"> • Update sequence number (USN) counter • Up-to-dateness vector • High-watermark

These mechanisms are implemented by the replication system in a sequence of events that occurs between two domain controllers.

The following diagram shows a simplified version of the sequence between source and destination domain controllers when the source initiates replication by sending a change notification.

Replication Sequence



[Back to Top](#)

Active Directory Data Updates

When a change is made to an object in a directory partition, the value of the changed attribute or attributes must be updated on all domain controllers that store a replica of the same directory partition. Domain controllers communicate data updates automatically through Active Directory replication. Their communication about updates is always specific to a single directory partition at a time.

Active Directory data is logically partitioned so that all domain controllers in the forest do not store all objects in the directory. Active Directory objects are instances of schema-defined classes, which consist of named sets of attributes. Schema definitions determine whether an attribute can be administratively changed. Attributes that cannot be changed are never updated and therefore never replicated. However, most Active Directory objects have attribute values that can be updated.

Different categories of data are stored in replicas of different directory partitions, as follows:

- Domain data that is stored in domain directory partitions:
 - Every domain controller stores one writable domain directory partition.
 - A domain controller that is a global catalog server stores one writable domain directory partition and a partial, read-only replica of every other domain in the forest. Global catalog read-only replicas contain a partial set of attributes for every object in the domain.
- Configuration data: Every domain controller stores one writable configuration directory partition that stores forest-wide data controlling site and replication operations.
- Schema data: Every domain controller stores one writable schema partition that stores schema definitions for the forest. Although the schema directory partition is writable, schema updates are allowed on only the domain controller that holds the role of schema operations master.
- Application data: Domain controllers that are running Windows Server 2003 can store directory partitions that store application data. Application directory partition replicas can be replicated to any set of domain controllers in a forest, irrespective of domain.

Changes to Attributes

Active Directory updates originate on one domain controller (originating updates) and the same update is subsequently made on other domain controllers during the replication process (replicated updates).

Object update behavior is consistent and predictable: when a set of changes is made to a specific directory partition replica, those changes will be propagated to all other domain controllers that store replicas of the directory partition. How soon the changes are applied depends on the distance between the domain controllers and whether the change must be sent to other sites.

The following key points are central to understanding the behavior of Active Directory updates:

- Changes occur at the attribute level; only the changed attribute value is replicated, not the entire object.
- At the time of replication, only the current value of an attribute that has changed is replicated. If an attribute value has changed multiple times between replication cycles (for example, between scheduled occurrences of intersite replication), only the current value is replicated.
- The individual values of a multivalued attribute are replicated separately under the following conditions:
 - The forest functional level is Windows Server 2003 interim or Windows Server 2003.
 - The attribute is linked.

Note

- Linked attributes have the following characteristics:
 - The attribute is multivalued.
 - The attribute has distinguished name syntax.
 - The attribute is marked as linked in the schema.
- An attribute is available for replication as soon as it is written.
- Replication is store-and-forward and moves sequentially through a set of connected domain controllers that host directory partition replicas.
- 8Multimaster conflict resolution is effective without depending on clock synchronization.
- Originating updates to a single object are written to the database in the same transaction, so partially written objects are not possible and a consistent view of the object is maintained.
- For replicated updates to large numbers of values in linked multivalued attributes, such as the **member** attribute of a group, updates are not always guaranteed to be applied in the same transaction. In this case, the updates are guaranteed to be applied in one or more subsequent transactions in the same replication cycle (all updates from one source are applied at the destination).
- After a replication cycle is initiated, all available changes to a directory partition on the source domain controller are sent to the destination domain controller, including changes that occur while the replication cycle is in progress.

Effect of Schema Changes on Replication

Attribute definitions are stored in **attributeSchema** objects in the schema directory partition. Changes to **attributeSchema** objects block other replication until the schema changes are performed. During replication of any directory partition other than the schema directory partition, the replication system first checks to see whether the schema versions of the source and the destination domain controllers are in agreement. If the versions are not the same, the replication of the other directory partition is rescheduled until the schema directory partition is synchronized.

Prior to upgrading a domain controller from Windows 2000 Server to Windows Server 2003, you must update the schema to be compatible with Windows Server 2003. When you run Adprep.exe, Windows Server 2003 schema is installed in the forest. This process upgrades the schema on each Windows 2000-based domain controller. Thereafter, you can begin upgrading domain controllers to Windows Server 2003.

Important

- The Windows Server 2003 schema update adds 25 indexed attributes to the schema directory partition. An update of this size can cause replication delays in a large database. For this reason, domain controllers that are running Windows 2000 Server must be running at a minimum Windows 2000 Service Pack 2 (SP2) plus all additional Windows updates. However, it is highly recommended that you install Windows 2000 Service Pack 3 (SP3) on all domain controllers prior to preparing your infrastructure for upgrade to the Windows Server 2003 operating system.

Replication of Attributes with Multiple Values

The smallest change that can be replicated in Windows Server 2003 Active Directory is a separate value in a multivalued attribute that is linked. The smallest change that can be replicated in Windows 2000 Active Directory is an entire attribute; even if the attribute is linked and multivalued, all values replicate as a single change.

Linked and Non-Linked Attributes

A linked attribute represents an interobject distinguished-name reference with multiple values. For example, the **member** attribute of a group object can have multiple values of user names, computer names, and other group object names. The **memberOf** attribute of a user, computer, or group object can have multiple values of more than one group name. The relationships between a group and its members are stored in a separate table in the directory database as link pairs. Because the **member** and the **memberOf** attributes are linked in the database and indexed for searching, the directory can be examined for all records in which the link pair is **member/memberOf** and the **memberOf** attribute identifies the group (for example, "What user objects have group X as a value in their **memberOf** attribute?").

Note

- Other linked multivalued attributes have distinguished name values, but the **member** attribute has the most significant effect on Active Directory administration and replication.

Attributes are marked in the schema as being linked. Only attributes with the distinguished name syntax **Object(DS-DN)** can be linked, but not all such attributes are linked. Non-linked distinguished-name attributes reference other objects in the same way as linked attributes do except that their behavior is different when a referred-to object is deleted, as described in "Replication of Deletion Updates" later in this section. In addition, non-linked attributes have a limit of 800 values. For attributes of this maximum size, there are no storage or replication drawbacks or limitations.

Group Membership Replication in Windows 2000 Forests

In a Windows 2000 forest, the attribute is the smallest unit of replication. Group membership is stored in the single, linked, multivalued member attribute of the group object. Because an originating update must be written in a single database transaction, and because the practical limit for a single transaction is 5,000 values, membership of more than 5,000 values is not supported in Windows 2000 Active Directory (a change to a single member results in a database write of all members). A group of this size represents a limitation both in terms of the database write operation that is required to record a change to an attribute of that size and the transfer of that much data over the network.

These conditions have the following impacts on replication, most notably for group and distribution list objects:

- **Lost changes:** If values of the same multivalued attribute are updated on two different domain controllers during a period of replication latency, the most recently changed replica of the attribute with all its multiple values is replicated and any earlier changes are lost. Changes to the separate values are not merged.

Note

- Because all changes to an object must be written in the same database transaction, multiple changes to a single group object can take a relatively long time to be written, which increases the likelihood of another change occurring to the same object prior to the completion of the original write.
- **Excessive network bandwidth consumption:** For example, when one member is added to a group of 3,000 members, the **member** attribute with all 3,001 values is transmitted between domain controllers. Transmission of all values to apply a change to only the updated value or values is inefficient use of network resources.

These limitations are effectively removed in a forest that has a forest functional level of Windows Server 2003 or Windows Server 2003 interim. At these levels, changes to the Active Directory replication system accommodate replication of individually updated **member** values, not the entire **member** attribute.

Group Membership Replication in Windows Server 2003 Forests

In a Windows Server 2003 forest that has a forest functional level of Windows Server 2003 or Windows Server 2003 interim, changes to certain multivalued attributes — those that have distinguished name values and are also linked — can be replicated on a per-value basis instead of the entire attribute being replicated. This type of replication is called linked-value replication. This feature is most significant where group membership is concerned.

Replication of individual **member** values, rather than the entire **member** attribute, provides the following benefits:

- Removed likelihood of losing entire sets of changes to the same group membership made on different domain controllers.
- Greatly reduced likelihood of update collisions, where the same **member** value is changed on different domain controllers at the same time and one update is lost.
- Improved network efficiency by transmitting only updated values and not the entire set of attribute values, which can include many thousands of values.
- Improved ability to transmit all attribute changes to a single object in the same update transaction.

Although replication of many thousands of individual membership updates can be accommodated in a Windows Server 2003 forest, LDAP writes have a practical limit of approximately 5,000 updates in a single transaction. Because originating updates are required to complete in a single transaction, a practical limit of approximately 5,000 updates to a single object is recommended.

Note

- Only originating updates must be applied in the same database transaction. Replicated updates can be applied in more than one database transaction.

Effect of Raising the Forest Functional Level on Existing Linked, Multivalued Attributes

Existing linked, multivalued attributes are not directly affected when you raise the forest functional level to enable linked-value replication. These attribute values are converted to replicate as single values only when they are modified. This design avoids the performance effects that would potentially result from rewriting the existing **member** attribute values of all group objects in the forest at the same time.

Because the **member** attribute is not converted until it is modified, a group that exceeded the 5,000-member limit in Windows 2000 continues to represent a replication issue because the original set of members continues to replicate as a unit under the new forest functional level. New members that are added and any member values that are updated replicate separately thereafter. Therefore, if the groups that were created in Windows 2000 do not exceed the 5,000-member limit, no replication issues are associated with the group.

For more information about how linked multivalued attributes are stored, see "[How the Data Store Works.](#)"

Originating Updates: Initiating Changes

As a Lightweight Directory Access Protocol (LDAP) directory service, Active Directory supports the following four types of update requests:

- Add an object to the directory.
- Modify (add, delete, or replace) attribute values of an object in the directory.
- Move an object by changing the name or parent of the object.
- Delete an object from the directory.

Each LDAP request generates a separate write transaction. An LDAP directory service processes each write request as an atomic transaction; that is, the transaction is either completed in full or not applied at all. The practical limit to the number of values that can be written in one LDAP transaction is approximately 5,000 values added, modified, or deleted at the same time.

A write request either commits and all its effects are durable, or it fails before completion and has no effect. A write request that commits is called an originating update. An originating update is initiated and committed at a specific replica. The absolute success or failure of an update applies even for requests that might affect several attributes of a single object, such as Add or Modify. In this case, if one attribute update fails, they all fail and the object is not updated.

An originating update enforces schema restrictions, including allowable parent object types and syntax for mandatory and optional attributes for an object. The restrictions are enforced according to the schema that exists on the domain controller at the moment of the update.

Originating Add

An Add request makes a new object with a unique **objectGUID** attribute. The values of all replicated attributes that are set by the Add request are stamped Version = 1.

The Add request fails immediately if the parent object does not exist or if the originating domain controller does not contain a writable replica of the parent object's directory partition.

Originating Modify

All Modify operations replace the current value of an attribute with a new value. A modify request can specify one of the following:

- That an attribute be deleted from the object. Attribute deletion is best thought of as replacing the attribute value with NULL. The NULL value occupies no storage of its own but does carry a stamp, as does any value that is stored as a directory attribute.
- That a value be added to the current value of an attribute, as when modifying an attribute that can have multiple values. The effect is to replace the current values with the current values plus the added value.

For each attribute in the request, a Modify request compares the new value in the request with the existing value. If the values are the same, the request to modify that attribute is ignored. If the resulting Modify request does not change any attributes of the object, the entire request is ignored.

Otherwise, a Modify request computes a stamp in the metadata for each new replicated attribute value by reading the version from the existing value (version=0 for an attribute that has never been written) and then adding 1 to this value. The Modify request replaces the old stamp values with new stamp values.

Originating Move

A Move request is essentially a special Modify request for a single attribute, the **name** attribute. The operation proceeds as described for the Modify request.

Originating Delete

A Delete request is essentially a special Modify request that does the following series of operations:

1. Sets the **isDeleted** attribute to TRUE, which marks the object as a tombstone (an object that has been deleted but not fully removed from the directory).
2. Changes the relative distinguished name of the object to a value that cannot be set by an LDAP application (a value that is impossible).
3. Strips all attributes that are not needed by Active Directory. A few important attributes (including **objectGUID**, **objectSid**, **distinguishedName**, **nTSecurityDescriptor**, and **uSNChanged**) are preserved on the tombstone.

Note

- Because these attributes are preserved, tombstones can be restored (reanimated) by applications that use the LDAP API for undeleting an object.
4. Moves the tombstone to the Deleted Objects container, which is a hidden container within those directory partitions that allow deletions.

On domain controllers that are running Windows Server 2003, you can restore tombstones in the Deleted Objects container to an active state.

Configuration Objects Protected from Originating Deletion

Certain objects in the configuration directory partition are critical to the functioning of a domain controller. These objects and their child objects are protected from deletion. Each domain controller has its own set of protected objects, as follows:

- The NTDS Settings (class **nTDSDSA**) object that represents a domain controller in the replication topology.
- The cross-reference (class **crossRef**) objects that represent the writable directory partitions stored on the domain controller.
- The RID object for the domain controller.

These objects are *protected* at that domain controller, as follows:

- The domain controller rejects an originating deletion of a protected object.
- The domain controller does not carry out a replicated deletion of a protected object. Instead, the threatened protected object is revived by updating its replication metadata as if each attribute had just been updated.

Protected objects can become deleted when the domain controller that stores them is removed from service as a domain controller by removing Active Directory. Protection of these objects relies on a replication mechanism that ensures that a revived object is replicated out rather than remaining only on the domain controller that protects it.

For example, if you remove a domain controller from a site, the NTDS Settings object disappears from view in Active Directory Sites and Services. However, Active Directory preserves replication metadata for the object as if the object still exists. Domain controllers that ordinarily replicate from the missing domain controller continue to attempt replication.

Therefore, if you reinstate a domain controller, the NTDS Settings object reappears and replication continues immediately as if the object had been there all along. Otherwise, replication could not begin until the new NTDS Settings object had replicated out to all other domain controllers in the forest.

Preservation of the NTDS Settings Object

The period of time during which the replication metadata of the NTDS Settings object is maintained is determined by an attribute of the Directory Service object (**cn=Directory Service,cn=Windows NT,cn=Services,cn=Configuration,dc=DomainName,dc=ForestRootDomainName**). This attribute, **replTopologyStayOfExecution**, has a default value of 14 days and a maximum value of half the tombstone lifetime.

For example, if you set the tombstone lifetime to 30 days and the **replTopologyStayOfExecution** value to 20 days, the actual stay-of-execution value is 15 days.

The root object of each directory partition replica has a multivalued attribute **repsFrom** that contains configuration and persistent state information associated with inbound replication from each source replica of that directory partition. The Knowledge Consistency Checker (KCC) is the replication topology component that

manages creation of connection objects. When the KCC detects a **repsFrom** value with no corresponding connection object, as is the case when the NTDS Settings object has been deleted, the KCC checks the tombstone for the NTDS Settings object for the time it was deleted and compares that time to the interval in **replTopologyStayOfExecution**. If the NTDS Settings object has been deleted for a time that is equal to or greater than the value in **replTopologyStayOfExecution**, then the KCC removes the **repsFrom** value and replication is no longer attempted with the deleted server.

Prior to the stay-of-execution lifetime, you can see evidence of failed replication attempts in Windows Support tools such as Dcdiag.exe and Ldp.exe. When these attempts are from a server designated as <unknown>, they are likely due to the stay-of-execution mechanism.

Replicated Updates: Propagating Changes

A replicated update is performed on one domain controller as a result of receiving replication of an originating update that was performed at another domain controller. There is not necessarily a one-to-one correspondence between originating and replicated updates. A single replicated update might reflect a set of originating updates (even updates originating at different domain controllers) to the same object.

For example, the manager of a user object can be changed at one domain controller at the same time the address of the same user is changed at another domain controller. A third domain controller might receive these changes separately to the user object and in turn replicate the changes to a fourth domain controller in a single replicated update.

Replication of Deletion Updates

Object deletions are replicated by replicating tombstones. After an object is deleted but before it is removed from the directory, object references that formerly referred to the object now refer to the deleted object's tombstone. The **isDeleted** attribute, which has a value of TRUE when an object is a tombstone, indicates the object deletion to other domain controllers. Deleted objects are stored in the DeletedObjects hidden container. Every directory partition has a DeletedObjects container.

Note

- Objects are moved to the DeletedObjects container according to system flags. Certain protected configuration objects, such as NTDS Settings, do not move to the DeletedObjects container when deleted.

By default, tombstones have a lifetime of 60 days, after which they are permanently removed from the directory database through a process called garbage collection.

The tombstone lifetime can be changed, but it is important to ensure that the tombstone lifetime is larger than the worst possible replication latency for any directory partition so that a tombstone cannot be deleted before it has replicated to every directory partition replica. In addition, Active Directory does not allow data to be restored from a backup image that is older than the tombstone lifetime.

Note

- A tombstone is invisible to normal LDAP searches. However, a tombstone is visible to searches that use the special LDAP control 1.2.840.113556.1.4.417.

When an object that has a linked attribute is deleted, the referencing link value is deleted immediately. Although the object itself becomes a tombstone (for example, a user object is deleted), the referent object (in this case, the group object) does not reference the tombstone of the deleted object. Rather, the group-user link value is simply deleted from the database and the group object shows no evidence of the former user's membership.

Replication of Absent Linked Object References

A different condition occurs when the object that the link refers to is simply removed as a referent, as when a user is removed from a group.

In a Windows 2000 forest, this condition results in replication of the entire linked **member** attribute. In a forest with a functional level of Windows Server 2003 or Windows Server 2003 interim, this condition is replicated on a per-value basis. In this case, the user value is marked "absent" in the database so that its condition can be replicated, much like a tombstone.

The deletion of the user object from the group replicates as an absent value in the **member** attribute of the group. After a tombstone lifetime, absent values are physically removed from the database. Until that time, the link value remains in the database. Such absent values can be restored prior to the end of the tombstone lifetime. If you add the user back to the group prior to the end of a tombstone lifetime, the link value is restored in the database by the value being marked present.

Deleting Group Objects in a Windows Server 2003 Forest

In a forest that has a forest functional level of Windows Server 2003 or Windows Server 2003 interim, you can delete a group object of any size and its multivalued **member** attribute values are cleaned up in the background. For this reason, it is not required for the deletion to complete in one transaction, as it is in a Windows 2000 forest. However, if you delete more than 5,000 members in the same transaction without

deleting the group itself, the same long-running transaction conditions can occur as described in "Group Membership Replication in Windows 2000 Forests" earlier in this section.

Therefore, if you must add, modify, or delete more than 5,000 members of the same group, do so in blocks of less than 5,000 members to avoid long-running transactions and excessive network bandwidth consumption.

For more information about linked and non-linked values, garbage collection, and tombstones, see "[How the Data Store Works](#)."

Deletions on Non-Replicating Domain Controllers

If a domain controller fails to replicate for a number of days that exceeds the tombstone lifetime, replicas of objects that have been deleted from a writable partition might remain in that domain controller's directory. Because the tombstones of the deleted objects are permanently removed from the directory at the end of the tombstone lifetime, a domain controller that fails to replicate changes for tombstoned objects never deletes them.

This condition can occur for a variety of reasons including:

- Prolonged misconfigurations.
- Prolonged errors in name resolution, authentication, or the replication engine that block inbound replication.
- Turning on a domain controller that has been offline for longer than the tombstone lifetime (default 60 days).
- Advancing system time or reducing tombstone lifetime values in an attempt to accelerate garbage collection before end-to-end replication has taken place for all directory partitions in the forest.

The condition of outdated objects can also occur due to hardware and software problems that render the domain controller unreachable. Regardless of the reason, a deleted object can remain on a domain controller any time the domain controller goes offline prior to receiving a deletion and remains offline for longer than the tombstone lifetime of that deletion.

These outdated objects, called lingering objects, create inconsistency in the directory. If a change is made to an outdated object on the reconnected domain controller, it is possible for the object to be recreated in the directory under certain conditions. To avoid this situation, replication of an outdated object is prohibited by default in newly created Windows Server 2003 forests.

Replication Consistency Setting

If the attributes on a lingering object never change, the object is never considered for replication. However, if an attribute changes, the attribute is considered for outbound replication. Because the destination domain controller does not hold the object for the attribute that is being replicated, an update cannot be performed. How this condition is resolved depends on the replication consistency setting on the domain controller.

A registry setting on domain controllers that are running Windows Server 2003 or Windows 2000 Server with SP3 provides a consistency value that determines whether a domain controller replicates and reanimates an updated object that has been deleted from all other replicas, or whether replication of such objects is blocked. The default settings are different on domain controllers that are running Windows 2000 Server with SP3 and Windows Server 2003.

Strict Replication Consistency

To avoid problems with reanimating objects that have been deleted, a domain controller that is running Windows Server 2003 in a newly created (not upgraded) Windows Server 2003 forest blocks inbound replication by default when it receives an update to an object that it does not have.

Note

- Active Directory replication uses update tracking to differentiate between replicating a newly created object and updating an attribute for an existing object. Replication of a lingering object is an attempt to update an attribute or attributes of an object that the destination domain controller cannot update because the object does not exist.

Replication is halted in the directory partition for the object until the lingering object is removed from the source domain controller or the strict replication consistency setting is disabled. For information about how lingering objects are removed, see "Lingering Object Removal" later in this section.

Loose Replication Consistency

When strict replication consistency is disabled, the effect is called "loose" consistency. By using loose consistency, the destination domain controller detects that it does not have the object for the attribute that is being replicated. The destination domain controller requests the entire object from the source partner, and thereby reanimates the object in its copy of the directory. The same process repeats on all domain controllers that do not have a copy of the object.

This mechanism can be used to cause lingering objects to be reanimated across the entire forest. If a lingering

object is discovered and its presence is intended, then perform any update to the object. As long as replication consistency is set to loose (strict replication consistency is disabled) on all domain controllers, the object will be reanimated as it replicates around the forest.

Loose replication consistency is the default setting for domain controllers that are running Windows 2000 Server with SP3 or later. The Windows 2000 Server default is not changed by upgrading to Windows Server 2003; strict replication consistency remains disabled and replication is allowed to proceed. Keeping the Windows 2000 Server setting is required to ensure that the upgraded domain and forest are consistent with Windows 2000 Server functionality. You must change the setting manually following the upgrade.

Storage for Consistency Setting

The setting for replication consistency is in the registry on each domain controller.

The value for the consistency setting is stored in the **Strict Replication Consistency** entry in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters.

The values are as follows:

- Value: 1 (Set to 0 to disable)
- Default: 1 (enabled) in a new Windows Server 2003 forest, otherwise 0.
- Data type: REG_DWORD

Note

- Having **Strict Replication Consistency** set to 0 or unset is equivalent to the Windows 2000 Server setting applied by the **Correct Missing Objects** registry entry. However, the semantics for **Correct Missing Objects** are the opposite of **Strict Replication Consistency**: **Correct Missing Objects**=1 is equivalent to **Strict Replication Consistency**=0 or unset.

Lingering Object Removal

On domain controllers that are running Windows Server 2003, you can use Repadmin to analyze and remove lingering objects from a domain controller that you suspect or know has not replicated for a tombstone lifetime or longer. If strict replication consistency is in effect and replication fails on the destination domain controller, event ID 1988 is logged in the Directory Service event log on the destination domain controller. Event ID 1988 indicates that the local domain controller has attempted to replicate an object from the source domain controller and the object is not present on the local domain controller because it might have been deleted and its tombstone already garbage collected. Event ID 1988 provides the GUID-based distinguished name of the source domain controller as well as the distinguished name and GUID of the outdated object. Replication of the directory partition containing the outdated object does not continue with the source domain controller until the situation has been resolved.

In this event, you can use an up-to-date domain controller as the authority against which to compare the objects on the source replication partner suspected of harboring lingering objects. This domain controller acts as the authoritative directory replica to reveal outdated objects in the suspect directory database on the destination.

In the Repadmin command-line arguments that remove lingering objects, the roles of source and destination are switched. The **repadmin /removelingeringobjects** command compares the directories of two domain controllers, as follows:

- A "source" domain controller that you designate as the authoritative server.
- A "destination" domain controller, which is the source replication partner that has attempted to replicate an outdated object.

To use the **repadmin /removelingeringobjects** command, both source and destination domain controllers must be running Windows Server 2003.

If the comparison reveals any mismatched objects, Repadmin either displays or removes the objects that are found on the destination but not on the source, depending on the arguments that you use. The advisory mode argument allows you to view the results of the command before you take action to remove any objects from the directory.

Repadmin /RemoveLingeringObjects Command Syntax

The command **repadmin /removelingeringobjects** has the following syntax:

```
/removelingeringobjects <Dest_DC_LIST> <Source DC GUID> <NC> [/ADVISORY_MODE]
```

where

- <Dest_DC_List> is the DNS or NetBIOS name of one or more domain controllers that you suspect of harboring lingering objects. <DC_List> provides the ability to target specific domain controllers, such as all domain controllers in a site, all global catalog servers, or domain controllers that hold specific operations master roles. To see the syntax for DC_List, type **repadmin /listhelp** at the command prompt.

- <Source DC GUID> is the GUID you obtained by running **repadmin /showrepl** against the source domain controller that you are using as the authoritative server.
- <NC> is the distinguished name of the directory partition that contains the lingering object.
- [/ADVISORY_MODE] is an optional switch that specifies that no deletions are performed on the destination domain controller, but are displayed (logged) only. Using this switch is recommended prior to allowing Repadmin to remove any objects.

To use this command, you must first obtain the GUID of the authoritative source domain controller by running **repadmin /showrepl <source_server>**, where <source_server> is the name of the domain controller that has a writable copy of the directory partition that will serve as the authoritative replica. The output of this command provides the "DC GUID," which the **/removelingeringobjects** command requires to identify the authoritative source.

RemoveLingerinObjects Implementation

When you run **repadmin /removelingeringobjects**, the tool performs the following steps to compare the directories of the source and destination domain controllers and log (or remove) any found lingering objects:

1. Check to ensure that the directory partition and the source domain controller are valid.
2. Verify that the user has the DS-Replication-Manage-Topology extended right on the directory partition container object specified in <NC>. This extended right is required to verify object state between two domain controllers. Members of the Domain Admins group have this right by default.
3. Ensure that both source and destination use the same objects for comparison by merging the up-to-dateness vectors to filter out any objects that have not replicated from the source to the destination or from the destination to the source. This check rules out a lingering object on the destination if the destination has not received the tombstone from the source, and vice versa. Any such non-replicated objects are removed from the comparison.
4. Create the list of object GUIDs for each domain controller to be compared. Examine the metadata of each object and use the merged up-to-dateness vector to determine whether the object should be present on both source and destination.
5. For each GUID that is in the list for the destination, determine if it is in the list of GUIDs for the source.
6. If a GUID is not found on the source, the object is identified to be outdated on the destination and is either displayed or deleted on the destination server. If advisory mode has been specified, the GUID is displayed only.

For more information about up-to-dateness vectors, see "Update Tracking by Domain Controllers" later in this section.

Update Tracking by Domain Controllers

Some directory services use timestamps to determine what changes need to be propagated, on the basis of preserving the last write. But keeping time closely synchronized in a large network is difficult. When the latest time of a directory write is the only means of determining which of two changes is recorded and replicated, skewed time on a domain controller can result in data loss or directory corruption.

Active Directory replication does not primarily depend on time to determine what changes need to be propagated. Instead it uses update sequence numbers (USNs) that are assigned by a counter that is local to each domain controller. Because these USN counters are local, it is easy to ensure that they are reliable and never run backward (that is, they cannot decrease in value).

Note

- Replication uses Kerberos v 5 authentication protocol for security, which does require that the time services on domain controllers are synchronized.

When a conflict occurs, instead of using timestamps as the primary mechanism to determine what updates are preserved, Active Directory uses volatility (number of changes) as the first element of the per-attribute stamps that are compared during conflict resolution. The second element is a timestamp. Therefore, if an attribute is updated once on domain controller A and once on domain controller B, the last writer's update is preserved. But if the attribute is updated on domain controller A, then on domain controller B, and then again on domain controller A, the update of domain controller A is preserved even if the clock of domain controller B is set forward from that of domain controller A. With Active Directory, clock skew can never prevent a value from being overwritten.

Domain controllers use USNs to simplify recovery after a failure. When a domain controller is restored following a failure, it queries its replication partners for changes with USNs greater than the USN of the last change it received from each partner.

You do not need to be concerned with USNs and their implications unless you experience problems with replication that require troubleshooting.

Server Object GUID (DSA GUID) and Server Database GUID (Invocation ID)

The server object that represents a domain controller in the Sites container of the configuration directory partition has a globally unique identifier (GUID) that identifies it to the replication system as a domain controller. This GUID is called the DSA (Directory System Agent) GUID. The DSA GUID is the GUID of the NTDS Settings object (class **nTDSDSA**). Its value is stored in the **objectGUID** attribute of the NTDS Settings object of the domain controller server object. The DSA GUID is used by domain controllers to locate replication partners. For more information about this process, see "Locating Replication Partners" later in this section.

The DSA GUID is created when Active Directory is initially installed on the domain controller and destroyed only if Active Directory is removed from the domain controller. The DSA GUID ensures that the DSA remains recognizable when a domain controller is renamed. The DSA GUID is not affected by the Active Directory restore process.

The Active Directory database has its own GUID, which the DSA uses to identify the database instance (version of the database). The database GUID is stored in the **invocationId** attribute on the NTDS Settings object. Unlike the DSA GUID, which never changes for the lifetime of the domain controller, the invocation ID is changed during an Active Directory restore process to ensure replication consistency. For more information about replication following a restore process, see "Active Directory Replication on a Restored Domain Controller" later in this section.

On domain controllers that are running Windows Server 2003, the invocation ID also changes when an application directory partition is removed from or added to the domain controller.

Determining Changes to Replicate: Update Sequence Numbers

A source domain controller uses USNs to determine what changes have already been received by a destination domain controller that is requesting changes. The destination domain controller uses USNs to determine what changes it needs to request.

The current USN is a 64-bit counter that is maintained by each Active Directory domain controller as the **highestCommittedUsn** attribute on the rootDSE object. At the start of each update transaction (originating or replicated), the domain controller increments its current USN and associates this new value with the update request.

Note

- The rootDSE (DSA-specific Entry) represents the top of the logical namespace for one domain controller. RootDSE has no hierarchical name or schema class, but it does have a set of attributes that identify the contents of a given domain controller.

The current USN value is stored on an updated object as follows:

- **Local USN:** The USN for the update is stored in the metadata of each attribute that is changed by the update as the local USN of that attribute (originating and replicated writes). As the name implies, this value is local to the domain controller where the change occurs.
- **uSNChanged:** The maximum local USN among all of an object's attributes is stored as the object's **uSNChanged** attribute (originating and replicated writes). The **uSNChanged** attribute is indexed, which allows objects to be enumerated efficiently in the order of their most recent attribute write.

Note

- When the forest functional level is Windows Server 2003 or Windows Server 2003 interim, discrete values of linked multivalued attributes can be updated individually (see "Group Membership Replication in Windows Server 2003 Forests" earlier in this section). In this case, there is a **uSNChanged** associated with each link in addition to the **uSNChanged** associated with each object. Therefore, updates to individual values of linked multivalued attributes do not affect the local USN, only the **uSNChanged** attribute on the object.
- **Originating USN:** For an originating write only, the update's USN value is stored with each updated attribute as the originating USN of that attribute. Unlike the local USN and **uSNChanged**, the originating USN is replicated with the attribute's value.

Destination domain controllers use the originating USN to track changes they have received from other domain controllers with which they replicate. When requesting changes from a source domain controller, the destination informs the source of the updates it has already received so that the source never replicates changes that the destination does not need.

Two values are used by source and destination domain controllers to filter updates when the destination requests changes from the source replication partner:

- **Up-to-dateness vector.** The current status of the latest originating updates to occur on all domain controllers that store a replica of a specific directory partition.
- **High-watermark (direct up-to-dateness vector).** The latest originating update to a specific directory partition that has been received by a destination from a specific source replication partner during the current replication cycle.

Both of these values specify the invocation ID of the source domain controller.

Attributes to Send for Replication: Up-to-Dateness Vector

The up-to-dateness vector is a value that the destination domain controller maintains for tracking the originating updates that are received from all source domain controllers. When a destination domain controller requests changes for a directory partition, it provides its up-to-dateness vector to the source domain controller. The source domain controller uses this value to reduce the set of attributes that it sends to the destination domain controller.

The up-to-dateness vector contains an entry for each domain controller that holds a full replica of the directory partition. The up-to-dateness vector values include the database GUID (invocation ID) of the source domain controller and the highest originating write (based on the USN) received from the source domain controller. If the up-to-dateness entry that corresponds to source domain controller X contains the USN *n*, the destination domain controller guarantees that it holds all updates to a specific directory partition that originated at domain controller X and that have an originating USN value of less than or equal to *n*.

If the destination already has an up-to-date value, the source domain controller does not send that attribute. If the source has no attributes to send for an object, it sends no information at all about that object.

At the completion of a successful replication cycle between two replication partners, the source domain controller returns its up-to-dateness vector to the destination, including the highest originating USN on the source domain controller. The destination merges this information into its up-to-dateness vector. In this way, the destination tracks the latest originating update it has received from each partner, as well as the status of every other domain controller that stores a replica of the directory partition.

Timestamp on Up-to-Dateness Vector in Windows Server 2003

On domain controllers that are running Windows Server 2003, the up-to-dateness vector includes a timestamp that represents the last time the local (destination) domain controller has completed a full replication cycle with the source domain controller. The replication cycle may have occurred directly (direct replication partner) or indirectly (transitive replication partner). The timestamp is recorded whether or not the local domain controller actually received any changes from the partner.

By examining the timestamps, a domain controller can quickly identify other domain controllers that are not replicating. Warning messages are posted to the event log on each domain controller when non-replicating partners are discovered (Event ID 1864 in the Directory Service event log).

Objects to Consider for Replication: High-Watermark (Direct Up-to-Dateness Vector)

The high-watermark, or direct up-to-dateness vector, is a value that the destination domain controller maintains during replication to keep track of the most recent attribute change that it has received from a specific source domain controller for an object in a specific directory partition. When sending changes to a destination domain controller, the source domain controller provides the changes in increasing order of **uSNChanged**. Although the **uSNChanged** values from the source domain controller are not stored on objects at the destination domain controller, the destination domain controller keeps track of the **uSNChanged** value of the most recent object that was successfully updated from the source domain controller for a specific directory partition. This USN is called the destination's high-watermark with respect to the directory partition and the source domain controller.

When requesting changes during a replication cycle, the destination provides the high-watermark value with each request to the source domain controller, which in turn uses this value to filter the objects that it considers for continuing replication to the destination. If the **uSNChanged** value of an object on the source domain controller is less than or equal to the high-watermark value of the destination domain controller, the object update has already been received by the destination domain controller and is therefore not replicated. The high-watermark serves to decrease the CPU time and number of disk I/O operations that would otherwise be required.


The up-to-dateness vector and the high-watermark are complementary filter mechanisms that work together to decrease replication latency. Whereas the high-watermark prevents irrelevant objects from being considered by the source domain controller with respect to a single destination, the up-to-dateness vector helps the source domain controller to filter irrelevant attributes (and entire objects if all attributes are filtered) on the basis of the relationships between all sources of originating updates and a single destination.

For a specific directory partition, a domain controller maintains a high-watermark value for only those domain controllers from which it requests changes, but it maintains an up-to-dateness vector entry for every domain controller that has ever performed an originating update, which is typically every domain controller that holds a full replica of the directory partition.

Multiple Paths Without Redundant Replication

Multiple replication paths can exist between a pair of domain controllers. Multiple paths provide fault tolerance and can reduce latency. However, when multiple paths exist, you might expect the same change to be sent along each path to a specific domain controller or that a change might replicate in an endless loop. Active Directory prevents these potential problems with multiple paths by using the up-to-dateness vector. The ability to eliminate redundancy is called "propagation dampening."

The following is an example of how ordinary replication occurs:

1. DC A updates a password attribute. In this example, the originating USN of the attribute is set to 3.
 2. Destination DC B requests changes from source DC A and sends its high-watermark and up-to-dateness vector to DC A.
 3. According to the high-watermark that was passed by DC B, source DC A examines one or more objects, one of which contains the changed password. When DC A encounters the changed password attribute, it proceeds as follows:
 - a. First, DC A finds that the originating directory system agent (DSA) of the password attribute is DC A.
-  **Note**
- The DSA is the server-side process that creates an instance of a directory service. The DSA provides access to the physical store of directory information located on a hard disk.
- b. Therefore, DC A reads the up-to-dateness vector supplied by DC B and finds that DC B is guaranteed to be up-to-date with updates that originated at DC A and that have an originating USN of less than or equal to 2.
 - c. DC A then finds that the originating USN of the password attribute is 3.
 - d. Because 3 is greater than 2, DC A sends the changed password attribute to DC B.

To illustrate propagation dampening, suppose that DC B had already received the password update from DC C, which had received it from DC A. In this case, the entry in the up-to-dateness vector of DC B for DC A would contain the USN value 3, not 2. Therefore, DC A would not send the changed password to DC B.

For information about viewing the replication metadata, see "[Active Directory Replication Tools and Settings](#)."

Multimaster Conflict Resolution Policy

Active Directory must ensure that all domain controllers agree on the value of the updated attribute after replication occurs. The general approach to resolving conflicts is to order all update operations (Add, Modify, Move, and Delete) by assigning a globally unique (per-object and per-attribute) stamp to the originating update. Thus each replicated attribute value (or multivalue) is stamped during the originating update and this stamp is replicated with the value.

Conflict Resolution Stamp

The stamp that is applied during an originating write has the following three components:

- The version is a number that is incremented for each originating write. The version of the first originating write is 1. The version of each successive originating write is increased by 1.
- The originating time is the time of the originating write, to a one-second resolution, according to the system clock of the domain controller that performed the write.
- The originating DC is the DSA GUID of the domain controller that performed the originating write.

When stamps are compared, the version is the most significant, followed by the originating time and then the originating DC. If two stamps have the same version, the originating time almost always breaks the tie. In the extremely rare event that the same attribute is updated on two different domain controllers during the same second, the originating DC breaks the tie in an arbitrary fashion.

Two different originating writes of a specific attribute of a particular object cannot assign the same stamp because each originating write advances the version at a specified originating domain controller. The originating time does not contribute to uniqueness. Replicated writes cannot decrease the version because values with smaller versions lose during conflict resolution. You can see all three components of the stamp in the output of the **repadmin /showobjmeta** command.

In the case of a conflict, the ordering of stamps allows a consistent resolution, as described in the following table.

Replication Conflict Resolution

Replication Conflict	Description	Resolution
Attribute value conflict	A Modify operation sets the value of an attribute. Concurrently, at another domain controller, a Modify operation sets the value of the same attribute to a different	The attribute value at all domain controllers is the value with the larger version number in its stamp.

	value.	
Add or Move under deleted parent, Delete non-leaf object	An Add or Move operation makes an object a child of parent object. Concurrently, at another domain controller, a Delete operation deletes the parent object.	At all domain controllers, the parent object is deleted and the child object is a child of the special LostAndFound container in the directory partition. Stamps are not involved in the resolution.
Relative distinguished name conflict	An Add or Move operation names a child object below a parent object. Concurrently, at another domain controller, an Add or Move operation names a different child of the same parent with the same child name, resulting in two child objects with identical relative distinguished name values below the same parent object.	The child object whose naming attribute has the larger version number in its stamp keeps its given name. The child object whose relative distinguished name attribute (for example, CN for most objects, OU for organizational units, DC for domain components) has the smaller version number in its stamp is named by the following convention: At all domain controllers, a system-assigned value that is unique to the conflicting name and cannot conflict with any client-assigned value is assigned to the child object. For example, if the relative distinguished name of a child object was "CN=ABC" before conflict resolution, its relative distinguished name after resolution is "CN=ABC*CNF:<GUID>", where "*" represents a reserved character, "CNF" is a constant that indicates a conflict resolution, and "<GUID>" represents a printable representation of the objectGUID attribute value.

Tracking Object Creation, Replication, and Change

The following series of diagrams illustrates the replication-related data for a single object and one of its attributes as it goes from creation through replication.

The following diagram shows the replication-related data for the user object when it is first created on domain controller DC1. Before the user object is created, the current USN for the domain controller is 4710. When the object is created, the local USN of 4711 is assigned to each attribute of the user object, and the current USN for the domain controller increments from 4710 to 4711. Because the object has not yet changed, the value of its **uSNChanged** attribute is the same as its **uSNCreated** attribute 4711.

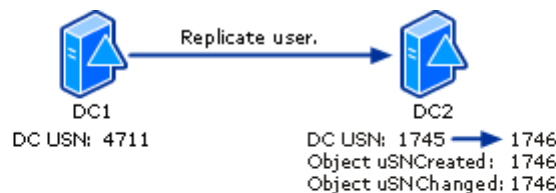
1. Replication-related Data on DC1 When a User Object is Created



Property	Value	Local USN	Version	Originating Time	Originating DC	Originating USN
cn	Jeff Smith	4711	1	2003-09-10 10:49.03	<DC1_GUID>	4711
userPassword	6Be8W5q-	4711	1	2003-09-10 10:49.03	<DC1_GUID>	4711
sAMAccountName	JSmith	4711	1	2003-09-10 10:49.03	<DC1_GUID>	4711
userPrincipalName	JSmith@contoso.com	4711	1	2003-09-10 10:49.03	<DC1_GUID>	4711

The next diagram shows the change to the destination domain controller when the new user object is replicated. The object is created as a replicated update on DC2. Notice that the per-attribute originating USN and stamp (version, originating time, originating DC) are replicated from DC1 to DC2, but the per-attribute local USN and per-object **uSNChanged** are unique to DC2.

2. Replication-related Data on DC2 When a New User Object is Replicated From DC1



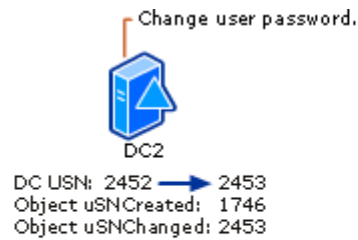
Property	Value	Local USN	Version	Originating Time	Originating DC	Originating USN
cn	Jeff Smith	1746	1	2003-09-10 10:49.03	<DC1_GUID>	4711
userPassword	6Be8W5q-	1746	1	2003-09-10 10:49.03	<DC1_GUID>	4711
sAMAccountName	JSmith	1746	1	2003-09-10 10:49.03	<DC1_GUID>	4711
userPrincipalName	JSmith@contoso.com	1746	1	2003-09-10 10:49.03	<DC1_GUID>	4711

The following information is transferred in the metadata of an updated attribute value from the source domain controller to the destination domain controller:

- The originating USN value for the updated attribute, which is the USN assigned by the domain controller on which the update was made.
- The stamp, which is used to resolve conflicts.

The next diagram illustrates the change in the replicated object on DC2 when someone changes the password (the **userPassword** property in the diagram) of the object on that domain controller. By this time, the current USN on DC2 has increased from 1746 to 2001. The update request changes the password and increments the current USN to 2002 on DC2. The request also sets the password attribute's originating USN and local USN to 2002 and creates a new stamp for the password value. The version number of this password's stamp is 2, which is one version number higher than the version of the previous password.

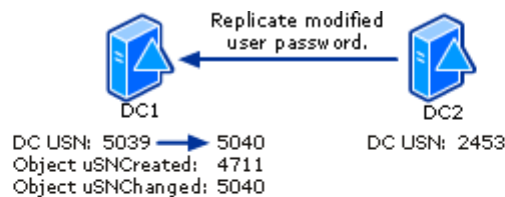
3. Replication-related Data on DC2 After the User Password Value Has Been Changed on DC2



Property	Value	Local USN	Version	Originating Time	Originating DC	Originating USN
cn	Jeff Smith	1746	1	2003-09-10 10:49.03	<DC1_GUID>	4711
userPassword	sEP3569?@2	2453	2	2003-09-10 11:53.29	<DC2_GUID>	2453
sAMAccountName	JSmith	1746	1	2003-09-10 10:49.03	<DC1_GUID>	4711
userPrincipalName	JSmith@contoso.com	1746	1	2003-09-10 10:49.03	<DC1_GUID>	4711

In the next diagram, the changed password is now replicated back to the original domain controller, whose current USN has increased to 5039. The replicated update increments the current USN of DC1 to 5040. The per-attribute originating USN and stamp (version, originating time, originating DC) are replicated from DC2 to DC1, and the per-attribute local USN and per-object **uSNChanged** values are set to 5040.

4. Replication-related Data on DC1 After the Password Change Has Replicated to DC1



Property	Value	Local USN	Version	Originating Time	Originating DC	Originating USN
cn	Jeff Smith	4711	1	2003-09-10 10:49.03	<DC1_GUID>	4711
userPassword	sEP3569?@2	5040	2	2003-09-10 11:53.29	<DC2_GUID>	2453
sAMAccountName	JSmith	4711	1	2003-09-10 10:49.03	<DC1_GUID>	4711
userPrincipalName	JSmith@contoso.com	4711	1	2003-09-10 10:49.03	<DC1_GUID>	4711

Active Directory Replication on a Restored Domain Controller

All domain controllers must be backed up routinely to ensure directory integrity. In case of failure on a domain controller, the backup media can be used to restore the domain controller to its state at the time of the backup. When a domain controller is restored from a backup, it can then be brought up-to-date by normal replication.

There are two general methods for restoring Active Directory from backup media, each of which has different replication consequences:

- Non-authoritative restore. Replication brings the domain controller up-to-date from its state at the time of backup, including updating deletions that have occurred since the time of the backup.
- Authoritative restore. Objects that were deleted can be reinstated.

Non-Authoritative Restore

Non-authoritative restore is the default method when performing a restore of Active Directory, and is used in the majority of restore situations, such as domain controller hard disk failure. A non-authoritative restore returns Active Directory on the domain controller to a state that is consistent with the time that the backup was taken. When the domain controller restarts following the restore process, it requests changes from its replication partners. Through the normal replication process, the restored domain controller receives any directory changes that have occurred since the time of the backup.

Because the restore process does not restore any previously deleted data to Active Directory, it is described as non-authoritative.

When a domain controller restarts after being restored from a backup, the domain controller must determine where to restart the USN sequence. If an existing USN is reused, serious replication problems could result. For example, suppose object A has USN=1000 and is replicated across the network. Three changes are made to object A, raising its version number to 4. A domain controller in the same domain is restored and creates a new object B with USN=1000. The version number of object B is 1, but it has the same USN as object A. For this reason, object B will never replicate because its version number is below that of object A.

To mitigate this risk, the Active Directory invocation ID is changed on the restored domain controller as part of the restore process. The effect is that the restored domain controller appears as a new replication partner to the domain controllers from which it pulls replication.

In the following example, three domain controllers (DC1, DC2, and DC3) exist as part of the Active Directory replication topology. The scenario includes the following steps:

Step 1

Prior to backup, DC1 maintains replication information about itself and its replication partners:

- Its own invocation ID (database GUID)
- Highest USN that it has committed for changes to the local database.
- Replication metadata for its two replication partners, DC2 and DC3:
 - Invocation ID of the local Active Directory database of each domain controller.
 - The up-to-dateness vector, as shown in the table below.

Step 2

Some changes have been made to the Active Directory database on DC1 since Step 1, which result in advancing the current USN on DC1 to 35. At this point, DC1 is backed up.

Step 3

A total of 14 additional change changes have been made to the local Active Directory database on DC1, advancing the highest committed USN on DC1 to 49. DC1 now begins replicating these changes to DC2. After replicating 12 of the 14 changes to DC2, DC1 becomes unavailable due to hard drive corruption. As a result, DC2 has USN 47 as the last change it received from DC1. At this point, the up-to-dateness vector on DC2 shows a highest committed USN of 47 for DC1.

Step 4

A non-authoritative restore has been completed and DC1 is restarted. During the restore process a new invocation ID has been assigned to the local Active Directory database on DC1. For replication purposes, DC1 adds its old invocation ID (GUID1) and highest committed USN of 35 to its up-to-dateness vector, effectively restoring it to its state as of Step 2.

Step 5

DC1 replicates in the 12 changes that were saved to DC2 before the restore (which corresponds to Step 3 in the table). The last two changes that occurred as originating updates on DC1 prior to going offline are lost.

DC1 Replication Metadata Values Pre- and Post-Non-Authoritative Restore

	Step 1: Prior to backup	Step2: Immediately post-backup	Step 3: Offline status before restore	Step 4: Immediately post-Restore	Step 5: After post-restore replication
Invocation ID of DC1	DC1-GUID1	DC1-GUID1	DC1-GUID1	DC1-GUID4	DC1-GUID4
Highest committed USN on DC1	20	35	49	35	47
Up-to-dateness vector on DC1	DC2-GUID2, USN=10; DC3-GUID3, USN=45	DC2-GUID2, USN=25; DC3-GUID3, USN=60	DC2-GUID2, USN=37; DC3-GUID3, USN=60	DC2-GUID2, USN=25; DC3-GUID3, USN=60; DC1-GUID1, USN=35	DC2-GUID2, USN=37; DC3-GUID3, USN=60; DC1-GUID1, USN=47

Authoritative Restore

The primary purpose of an authoritative restore is to reinstate objects that were deleted from Active Directory. To reinstate objects that were intentionally or accidentally deleted, a non-authoritative restore must be completed and followed by an authoritative restore.

The non-authoritative restore process cannot reinstate deleted objects from an older backup image because the backup media that was used to restore the domain controller contains an image of Active Directory that was created before the objects were deleted. In this case, the deletions would simply be replicated in from the up-to-date replication partner and applied by the restored domain controller. To reinstate a deleted object, an authoritative restore is required.

The authoritative restore process works as follows:

1. The domain controller is restarted in Directory Services Restore Mode and a non-authoritative restore of Active Directory is performed by using backup media that was created before the object was deleted.
2. Following the non-authoritative restore but prior to restarting, the object metadata is altered by using Ntdsutil.exe so that it has a higher USN than any other possible version of the object (by default, the version number is increased by 100,000). The effect is to render the object or objects as authoritative and reinstates them in Active Directory.

The authoritative restore process does not affect objects that were created after the backup was created.

Note

- Only the domain and configuration directory partitions can be marked as authoritative. The schema directory partition cannot be authoritatively restored.

[Back to Top](#)

Domain Controller Notification of Changes

Replication within a site occurs as a response to changes. On its NTDS Settings object, the source domain controller stores a **repsTo** attribute that lists all servers in the same site that pull replication from it. When a change occurs on a source domain controller, it notifies its destination replication partner, prompting the destination domain controller to request the changes from the source domain controller. The source domain controller either responds to the change request with a replication operation or places the request in a queue if requests are already pending. Replication occurs one request at a time until all requests in the queue are processed.

Note

- Replication between sites occurs according to a schedule, where the destination requests changes at the specified time. By default, change notification is not enabled on site links, although this setting can be changed.

Notification Delay Values

When a change occurs on a domain controller within a site, two configurable intervals determine the delay between the change and subsequent events:

- **Initial notification delay.** The delay between the change to an attribute and notification of the first partner. This interval serves to stagger network traffic caused by intrasite replication. When a domain controller makes a change (originating or replicated) to a directory partition, it starts the timer for the interval; when the timer expires, the domain controller begins to notify its replication partners (for that directory partition and within its site) that it has changes. The default value is 15 seconds.
- **Subsequent notification delay.** Notification of the first partner and notification of each subsequent partner. A domain controller does not notify all of its replication partners at one time. By delaying between notifications, the domain controller distributes the load of responding to replication requests from its partners. The default delay between notifications is 3 seconds.

The default values for the initial and subsequent notification delay intervals depend variably on the version of the operating system, the upgrade path, and the forest functional level.

The default initial notification delay is 15 seconds and the subsequent notification delay is 3 seconds on a domain controller under any of the following conditions:

- The forest functional level is Windows Server 2003 and the default initial notification delay value was in effect on the domain controller if it was upgraded from Windows 2000. If non-default values are set before the upgrade, the non-default value is retained.
- The domain controller has been created from a new installation of Windows Server 2003 (not upgraded) in a Windows 2000 or Windows Server 2003 forest.

- The domain controller has been upgraded directly from Windows NT 4.0 to Windows Server 2003.

Initial notification delay is 300 seconds and subsequent notification delay is 30 seconds under either of the following conditions:

- The domain controller is running Windows 2000 Server.
- The domain controller has been upgraded from Windows 2000 to Windows Server 2003 and the forest functional level is Windows 2000.

Storage of Intrasite Notification Delay Values

On a domain controller that is running Windows Server 2003, intrasite notification delay values are specific to each directory partition and are stored in two attributes of the cross-reference object for each directory partition.

Windows Server 2003 Storage of Notification Delay Values

On domain controllers that are running Windows Server 2003, the attributes that store the change notification values are located on each cross-reference object in the Partitions container within the configuration directory partition:

- The value for initial change notification delay is stored in the **msDS-Replication-Notify-First-DSA-Delay** attribute.
- The value for subsequent notification delay is stored in the **msDS-Replication-Notify-Subsequent-DSA-Delay** attribute.

These attributes do not exist in Windows 2000 Server.

Although the attribute values are present on all domain controllers that are running Windows Server 2003, the default values of 15 seconds for initial notification delay and 3 seconds for subsequent notification delay are in effect only under the conditions described in "Default Notification Delay Values" earlier in this section.

Windows 2000 Server Storage of Notification Delay Values

On domain controllers that are running Windows 2000, notification delay values are stored in registry entries on each domain controller. The registry entries are as follows:

- The value for the delay before the first change notification is stored in the **Replicator notify pause after modify (secs)** entry in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters.
The default value is 300 seconds.
- The value for the delay before each subsequent change notification is stored in the **Replicator notify pause between DSAs (secs)** entry in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters.
The default value is 30 seconds.

Transfer of Registry Notification Delay Values on Windows Server 2003 Upgrade

Notification delay values for first and subsequent change notification are transferred from Windows 2000 registry settings during upgrades to Windows Server 2003 as follows:

- If the default registry value has been changed on a domain controller running Windows 2000, the registry entry and its value are preserved when the domain controller is upgraded to Windows Server 2003.
- If the default registry value has not been changed on the domain controller running Windows 2000, the entry is deleted from the registry when the domain controller is upgraded to Windows Server 2003. For newly installed domain controllers (not upgraded), the registry entries do not exist.

Note

- The registry entries can be added to the registry to override the cross-reference value, if needed, to control notification by a specific Windows Server 2003-based domain controller.

Notification Delay Values and Their Application by Domain Controllers

To accommodate both locations of notification delay information (in the registry on domain controllers that are running Windows 2000 and in the attribute on the cross-reference object on domain controllers that are running Windows Server 2003), the process that is used to determine change notification values considers all possibilities, favoring the setting in the registry if it exists, as follows:

1. Windows Server 2003 and Windows 2000 domain controllers: assume the default values of 300 seconds and 30 seconds for the first notification delay and subsequent notification delays, respectively.
2. Windows Server 2003 only: check the cross-reference object for the directory partition to which the

change has occurred. If a value is set, use this value.

3. Windows Server 2003 and Windows 2000: check the domain controller's registry for the presence of the respective registry values and respond according to forest functional level, as follows:
 - Windows 2000 operating system and Windows 2000 forest functional level: If a value is set, use this value instead of the default value.
 - Windows Server 2003 operating system, Windows 2000 forest functional level: If a value is set, use this value instead of the default value.
 - Windows Server 2003 operating system, Windows Server 2003 forest functional level: If a value is set, use this value to override the value on the cross-reference object for all directory partitions.

[Back to Top](#)

Identifying and Locating Replication Partners

Replication occurs in one direction between two domain controllers at a time. The replication topology determines the replication partnerships between source and destination domain controllers. As a replication source, the domain controller must determine the replication partners it must notify when changes occur. As a replication destination, the domain controller participates in replication either by responding to notification of changes from a source, or by requesting changes to initiate replication when it starts up or in response to a schedule.

Destination Identification of Source Replication Partners

When the KCC creates the replication topology, it creates connection objects on destination domain controllers that represent the inbound connection from the replication source domain controller. For each source domain controller that is represented by an inbound connection object, the KCC writes information to the **repsFrom** attribute of the directory partition object for each directory partition that the destination domain controller has in common with the source domain controller. This information is local to the destination domain controller and is not replicated. For a destination domain controller, the **repsFrom** attribute contains the following information about each source domain controller:

- Directory partition name.
- NTDS Settings object distinguished name.
- GUID-based DNS name (CNAME alias).
- Whether replication is intrasite, intersite using IP (RPC), or intersite using SMTP.
- Flags that govern the options, which indicate whether notification is allowed (no value is present) or not allowed (NEVER_NOTIFY is present).
- Invocation ID, for the purpose of detecting a change due to a restore from backup.
- Direct USN vector (high-watermark), which is the USN pair of object USN (OU) and property USN (PU).
- Time of last attempt and time of last success.
- Last error code.
- Count of consecutive failures, if any.

If notification is enabled on the destination domain controller, the destination responds to notification messages from source domain controllers by sending a request for changes. If notification is not enabled, the destination responds to notification requests with an error.

Notification is enabled on a destination domain controller except under the following conditions, in which the KCC sets the option NEVER_NOTIFY in **repsFrom**:

- The connection object that the KCC creates is inbound from a server in a different site. Such a destination is designated as a bridgehead server for the directory partition. Bridgehead servers pull replication according to a schedule on the site link object.
- The connection object is created to identify the source from which this domain controller replicates to add Active Directory as a new domain controller.

Source Identification of Destination Replication Partners

The source domain controller keeps track of the replication partners that pull changes from it and uses the information to locate partners for change notification. This information is not provided by the KCC, but rather by the source domain controller itself during a replication cycle. The first time a domain controller receives a request for changes from a new destination, the source creates an entry for the destination in the **repsTo** attribute on the respective directory partition object.

Whenever the source has changes, it sends a notification to all replication partners that are identified in the **repsTo** value for the respective directory partition. Like the **repsFrom** data, this information is stored locally on the domain controller and is not replicated. When updates occur, the source domain controller checks the **repsTo** attribute to determine the identities of its destination replication partners. The source domain controller notifies them one by one that changes are available.

Note

- The output of the command **repadmin /showrepl /v /all** shows the contents of the **repsFrom** value (INBOUND NEIGHBORS) and the **repsTo** value (OUTBOUND NEIGHBORS).

If a destination domain controller has the NEVER_NOTIFY option set, the destination responds to a notification message with an error. When the error is received by the source domain controller, it removes the entry for that destination from its **repsTo** data.

If a destination domain controller moves to a different site, the KCC adjusts the notify setting to indicate that notification is not needed.

Locating Replication Partners

The partner that initiates replication locates its partner by querying DNS to look up the IP address of the partner according to the GUID of the NTDS Settings object (class **nTDSDSA**). The NTDS Settings object represents the directory system agent (DSA) on the domain controller. Its GUID uniquely identifies the domain controller and is guaranteed to find the correct server, even if the name of the server has been changed.

The GUID of the NTDS Settings object is stored in the **objectGUID** attribute. The DSA GUID is created when Active Directory is installed on the domain controller, and is destroyed only if Active Directory is removed from the domain controller to create a member server.

Note

- The Active Directory database also has a GUID, which is used by the DSA to identify the specific versions of the database when a database has been restored. This GUID is stored in the **invocationId** attribute on the NTDS Settings object.

As part of the DNS registration process, the Net Logon service on every domain controller registers a canonical name (CNAME) resource record, or alias record, which is constructed using the DSA GUID as the alias and maps to the DNS fully qualified domain name (FQDN) of the respective domain controller. The format of CNAME alias is:

DsaGuid._msdcs.DNSForestName.

To initiate replication, the domain controller retrieves the GUID-based DNS name that is stored in **repsTo** (for a source domain controller that is sending a change notification) and **repsFrom** (for a destination domain controller that is initiating replication) and queries DNS for the CNAME resource record. DNS responds by returning both the CNAME resource record and the A resource record, which contains the IP address of the destination domain controller. The domain controller uses information in the CNAME resource record to authenticate to the replication partner. Therefore, by using the CNAME and A resource record data, the domain controller can initiate replication.

Note

- The *_msdcs.DnsForestName* DNS zone contains a variety of forest-wide service (SRV) resource records that are used to locate special servers, such as domain controllers and global catalog servers, and to facilitate replication. In the context of this discussion, it is important to note that if a DNS server that is authoritative for the *_msdcs.DnsForestName* zone is not available, replication between domain controllers cannot occur.

For more information about the DSA and application directory partitions, see "[How the Data Store Works](#)." For more information about SRV resource records and the *_msdcs* subdomain, see "[DNS Support for Active Directory Technical Reference](#)."

[Back to Top](#)

Urgent Replication

Certain important events trigger replication immediately, overriding existing change notification. Urgent replication is implemented immediately by using RPC/IP to notify replication partners that changes have occurred on a source domain controller. Urgent replication uses regular change notification between destination and source domain controller pairs that otherwise use change notification, but notification is sent immediately in response to urgent events instead of waiting the default period of 15 seconds (or 300 seconds on domain controllers that are running Windows 2000).

Events That Trigger Urgent Replication

Urgent Active Directory replication is always triggered by certain events on all domain controllers within the

same site. When you have enabled change notification between sites, these triggering events also replicate immediately between sites.

Between Windows Server 2003-based and Windows 2000-based domain controllers in the same site, immediate notification is caused by the following events:

- Assigning an account lockout, which a domain controller performs to prohibit a user from logging on after a certain number of failed attempts.
- Changing the account lockout policy.
- Changing the domain password policy.
- Changing a Local Security Authority (LSA) secret, which is a secure form in which private data is stored by the LSA (for example, the password for a trust relationship).
- Changing the password on a domain controller computer account.
- Changing the relative identifier (known as a "RID") master role owner, which is the single domain controller in a domain that assigns relative identifiers to all domain controllers in that domain.

Urgent Replication of Account Lockout Changes

Account lockout is a security feature that sets a limit on the number of failed authentication attempts that are allowed before the account is "locked out" from a further attempt to log on, in addition to a time limit for how long the lockout is in effect.

The PDC emulator receives urgent replication of account lockouts. In Active Directory domains, a single domain controller in each domain holds the role of PDC emulator, which simulates the behavior of a Windows NT version 3.x-based or Windows NT 4.0-based PDC. In Windows NT domains, the only domain controller that can accept updates is the PDC. If authentication fails at a BDC, the authentication request is passed immediately to the PDC, which is guaranteed to have the current password.

An account lockout is urgently replicated to the PDC emulator and is then urgently replicated to the following:

- Domain controllers in the same domain that are located in the same site as the PDC emulator.
- Domain controllers in the same domain that are located in the same site as the domain controller that handled the account lockout.
- Domain controllers in the same domain that are located in sites that have been configured to allow change notification between sites (and, therefore, urgent replication) with the site that contains the PDC emulator or with the site where the account lockout was handled. These sites include any site that is included in the same site link as the site that contains the PDC emulator or in the same site link as the site that contains the domain controller that handled the account lockout.

In addition, when authentication fails at a domain controller other than the PDC emulator, the authentication is retried at the PDC emulator. For this reason, the PDC emulator locks the account before the domain controller that handled the failed-password attempt if the bad-password-attempt threshold is reached.

Note

- When a bad password is used in an attempt to change a password, the lockout count is incremented on that domain controller only and is not replicated. As such, an attacker could try (of domain controllers)* (lockout threshold -1) + 1 guesses before the account is locked out. Although this scenario has a relatively small impact on account lockout security, domains with an exceptionally high number of domain controllers represent a significant increase in the total number of guesses available to an attacker. Because a user cannot specify the domain controller on which the password change is attempted, an attack of this type requires an advanced tool.

Replication of Password Changes

Password changes are replicated differently than both normal (non-urgent) replication and urgent replication. Changes to security account passwords present a replication latency problem wherein a user's password is changed on domain controller A and the user subsequently attempts to log on, being authenticated by domain controller B. If the password has not replicated from A to B, the attempt to log on fails. Active Directory replication remedies this situation by forwarding password changes immediately to a single domain controller in the domain, the PDC emulator.

In Active Directory, when a user password is changed at a domain controller, that domain controller attempts to update the respective replica at the domain controller that holds the PDC emulator role. Update of the PDC emulator occurs immediately, without respect to schedules on site links. The updated password is propagated to other domain controllers by normal replication within a site.

When the user logs on to a domain and is authenticated by a domain controller that does not have the updated password, the domain controller refers to the PDC emulator to check the credentials of the user name and password rather than denying authentication based on an invalid password. Therefore, the user can log on successfully even when the authenticating domain controller has not yet received the updated password. On domain controllers that are running Windows Server 2003 or Windows 2000 Server with SP4, if the authentication is successful at the PDC emulator, the PDC emulator replicates the password immediately to the

requesting domain controller to prevent that domain controller from having to check the PDC emulator again.

If the update at the PDC emulator fails for any reason, the password change is replicated non-urgently by normal replication.

For clients that are running Windows NT 4.0 or clients that are running Windows 95 or Windows 98 without the Directory Service Client Pack, the client attempts to contact the PDC emulator. If the client has the Directory Service Client Pack installed, the client contacts any domain controller and the contacted domain controller then attempts to contact the PDC emulator.

Note

- The Group Policy setting "Contact PDC on logon failure" can be disabled to keep a domain controller from contacting the PDC emulator if the PDC emulator role owner is not in the current site. If this setting is disabled, the password change reaches the PDC emulator non-urgently through normal replication.

[Back to Top](#)

Network Ports Used by Active Directory Replication

By default, RPC-based replication uses dynamic port mapping. When connecting to an RPC endpoint during Active Directory replication, the RPC run time on the client contacts the RPC endpoint mapper on the server at a well-known port (port 135). The server queries the RPC endpoint mapper on this port to determine what port has been assigned for Active Directory replication on the server. This query occurs whether the port assignment is dynamic (the default) or fixed. The client never needs to know which port to use for Active Directory replication.

Note

- An endpoint comprises the protocol, local address, and port address.

In addition to the dynamic port 135, other ports that are required for replication to occur are listed in the following table.

Port Assignments for Active Directory Replication

Service Name	UDP	TCP
LDAP	389	389
LDAP		686 (Secure Sockets Layer [SSL])
LDAP		3268 (global catalog)
Kerberos	88	88
DNS	53	53
SMB over IP	445	445

Replication within a domain also requires File Replication service (FRS) using a dynamic RPC port.

[Back to Top](#)

Related Information

The following resources contain additional information that is relevant to this section.

- [How the Data Store Works](#)
- [Active Directory Replication Topology Technical Reference](#)
- [Active Directory Schema Technical Reference](#)
- [DNS Support for Active Directory Technical Reference](#)