# Active Directory® Replication

This page left blank intentionally

## Lesson 1: Active Directory Replication Model

### Overview

The Active Directory® replication model encompasses the manner in which changes are propagated and tracked among domain controllers.

#### What You Will Learn

After completing this lesson, you will be able to:

* Explain key replication concepts
* Describe features such as application partitions and functional levels
* Describe the process by which an object is replicated between domain controllers
* Describe the purpose of the Knowledge Consistency Checker
* Explain variations in the replication process

#### Related Topics Covered in this Lesson

* File Replication Service
* Schema
* Flexible Single Master Operations

#### Recommended Reading

* Deployment Guide Resource Kit
* Distributed Systems Guide Resource Kit
* Microsoft® Corporation, Building Enterprise Active Directory Services - Notes from the Field, Microsoft Press, 2000. ISBN: 0-7356-0860-1

### Directory Partition Replicas

Slide Objective

Lead In



A directory partition replica can be a full (master) replica or a partial replica. A full replica contains all attributes of all directory partition objects and is both readable and writable. Each domain controller stores at least three full, writable directory partition replicas as follows:

* The schema partition, which contains all class and attribute definitions for the forest. There is one schema directory partition per forest.
* The configuration partition, which contains replication configuration information (and other information) for the forest. There is one configuration directory partition per forest.
* The domain partition, which contains all objects that are stored by one domain. There is one domain directory partition for each domain in the forest.

A full replica of a domain’s partition is stored on all domain controllers of that domain (and nowhere else); a full replica of a forest’s configuration and schema partitions is stored on all domain controllers of that forest.

A partial replica contains a subset of the attributes of all directory partition objects and is read-only. Partial replicas are stored only on Global Catalog servers. An attribute is contained in a partial replica if and only if the attribute’s *attributeSchema* object has *isMemberOfPartialAttributeSet* equal to TRUE.

On a domain controller, a single database stores copies of those objects that are pertinent to only that domain, in addition to copies of the schema and the configuration objects, which apply to all domains in the forest. On domain controllers that are Global Catalog servers, the database also stores partial replicas of directory partition objects from other domains. Partial replicas are stored on Global Catalog servers so that searches of the entire directory can be achieved without requiring referrals from one domain to another.

**Important:**

Note the difference between the directory tree (the forest) and the physical database on a specific domain controller in that forest. The directory includes all of the objects in the forest. The directory database on a specific domain controller in the forest includes replicas of the domain objects for only that domain in addition to the replicas of the configuration and schema objects for the entire forest.

### Application Partitions

Slide Objective

Lead In



Application partitions are another type of directory partition is available that can be hosted on domain controllers running Microsoft® Windows Server™ 2003. In addition to the configuration, schema, and domain directory partitions, a domain controller that is running Windows Server 2003 can store one or more application directory partitions, which can be created by applications or administrators as container objects. Application directory partitions can be used by applications to store application-specific data in a scope that is different than the domain or forest. For example, Windows Server 2003 DNS can use application directory partitions to store dynamically updated DNS zone data on only those domain controllers in the domain that run the DNS Server service.

Application partitions are not permitted to contain security principals as objects. Therefore, users, groups and computers cannot be placed in them. However, security can be defined on the objects that exist using access control lists (ACLs).

In most cases, an application partition is created by an application or service (such as DNS or TAPI). However, there may be a need to manually create a partition or replica member. This can be accomplished with NTDSUTIL. Unlike the other partition types, the contents an application partition are not replicated to the global catalog.

#### Credentials Required

The creation of new application directory partitions is much like the creation of child domains. You must be a member of the Enterprise Administrators group in order to create new application directory partitions. The addition of replicas also requires an enterprise administrator.

##### DNS Partitions

The domain naming service (DNS) application partitions (ForestDNSZones and DomainDNSZones) are unique because they are created by the DNS service, not by an administrator. The DNS server service runs under the Local System context to create both DNS partitions. Local System is a member of the Enterprise Domain Controllers group, which has permission to create application partitions as well as add replica members to them.

#### Windows Server 2003 Domain Naming Master

During the creation of an application partition or the addition of new application partition replica members, the Domain Naming master will be consulted. The Active Directory will not allow the attribute which stores the partition’s membership (msDS-NC-Replica-Locations) to be changed on any domain controller other than the Domain Naming master.

Because a Microsoft® Windows® 2000 domain controller does not recognize application partitions, any create or modify operations will fail if the Domain Naming master is not running Windows Server 2003. The failure may be reported by Ntdsutil or the DNS service during the attempt to create the partition, or by WINNT32 and DCPROMO.EXE when the first domain controller (forest-wide + domain-wide DNS partition) is added to the forest or any domain (domain-wide DNS).

**Note:**

Application partitions are sometimes referred to as “non-domain naming contexts” or NDNCs. This document and others may refer to application partitions as NDNCs.

### Replication Model Components

Slide Objective

Lead In



The following mechanisms contribute to the overall replication system:

* Multimaster loose consistency with convergence maintains data integrity.
* “Multimaster” which means that a directory partition can have many writable replicas, or copies, that must be kept consistent between domain controllers in the same forest. The replication system propagates changes made on any specific domain controller to all other domain controllers in the forest that store the directory partition in which the change occurs.
* “Loose consistency” which means that the replicas are not guaranteed to be consistent with each other at any particular point in time because changes can be applied to any full replica at any time. However, given enough time with no updates they will return to a consistent state.
* “Convergence” which means that if the system is allowed to reach a steady state in which no new updates are occurring and all previous updates have been completely replicated, all replicas are guaranteed to converge on the same set of values.
* Store-and-forward replication, which means that changes are not sent directly from one domain controller to all other domain controllers. Instead, a change is sent directly to only a subset of domain controllers. This subset of domain controllers is then responsible for sending the change to other domain controllers, and so on, until the change has reached every domain controller.
* Pull replication, which means that domain controllers request (pull) updates from replication partners. The domain controller in which a change originates does not “push” the change unsolicited to other domain controllers.
* State-based replication, which means that instead of storing a full change log, each directory partition replica stores per-object and per-attribute data to support replication.

### Multimaster Replication

Slide Objective

Lead In



True **multimaster** replication can be contrasted with Microsoft® Windows NT® 4.0 that uses a **master-slave** approach to updates where all updates must be made to the master copy of the directory and then be replicated to the slave copies. This system is adequate for a directory that has a small number of copies and for an environment where all of the changes can be applied centrally. But this approach does not scale beyond small-sized organizations nor does it address the needs of decentralized organizations. With Active Directory, no one domain controller is the master. Instead, all domain controllers within a domain are equivalent. Changes can be made to any domain controller, unlike a single-master system, where changes must be made to one server. In the single-master system, the primary server replicates the updated information to all other directory servers in the domain.

With multimaster replication, it is not necessary for every domain controller to replicate with every other domain controller. Instead, the system implements a robust set of connections that determines which domain controllers replicate to which other domain controllers to ensure that networks are not overloaded with replication traffic and that replication latency is not too long causing inconvenience to users. The set of connections through which changes are replicated to domain controllers in an enterprise is called the **replication topology**.

Multimaster update capability provides high availability of write access to directory objects because several servers can contain writable copies of an object. Each domain controller in the domain can accept updates **independently**, without communicating with other domain controllers. The system resolves any conflicts in updates to a specific directory object. If updates cease and replication continues, all copies of an object eventually reach the same value.

The manner in which a directory service stores information directly determines the performance and scalability of the directory service. Directory services must handle a large number of **queries** compared to the number of **updates** they must process. By creating multiple copies of the directory and keeping the copies consistent, the directory service can handle more queries per second.

Replication behaviour is **consistent and predictable**; given a set of changes to a specific replica, it can be predicted that changes are going to be propagated to all other replicas.

### Store-and-Forward Replication

Slide Objective

Lead In



Store-and-forward replication is designed to reduce communication over slow WAN links. An update replicates first to nearby replicas and from there to replicas that are farther away.

Store-and-forward replication eliminates the need to send every change directly from the server that is accepting the change to all other servers that hold replicas of the affected directory partition. In the slide, the domain controllers in each site communicate with servers in the same site but only DC1 and DC4 are responsible for communicating updates across the WAN link. Any changes made on DC2 or DC3 eventually arrive at site Paris, via DC1 and DC4. This assumes all the domain controllers are in the same domain. Store-and-forward greatly reduces the WAN traffic that is produced by replication.

To alleviate the administrative complexity of managing connections between all domain controllers, the system can create the topology automatically. Replication can be indirectly administered by defining a simplified network model within the directory. This model is based on concepts of sites, site links, and site link bridges. Based on this model, Active Directory creates replication connections that allow Active Directory to perform replication. When failures occur, Active Directory modifies replication connections to keep replication going. You also have the option of manually creating replication connections to exert finer control. Manually created connections coexist with automatically generated ones, so if you want to fine-tune one connection, you need not sacrifice the benefits of automatic management for other connections.

### Pull Replication

Slide Objective

Lead In



Active Directory uses *pull* replication. In pull replication, a *destination* replica requests information from a *source* replica. The request specifies the information that the destination needs, based on its knowledge of changes already received from the source and from all other domain controllers in the domain. When the destination receives information from the source, it applies that information, bringing itself more up-to-date. The destination’s next request to the source excludes the information that has already been received and applied.

The alternative is push replication. In push replication, a source sends information to an unsolicited destination, in an attempt to bring the destination more up-to-date. Push replication is problematical because it is difficult for the source to know what information the destination needs. Perhaps the destination has received the same information from another source. If a source sends information to a destination, there is no guarantee that the destination is going to apply it; if the source assumes otherwise, the system is unreliable.

### State-based Replication

Slide Objective

Lead In



Active Directory uses a state-based approach to replication. In **state-based** replication, each master in the multimaster system applies updates to its replica as they arrive, without maintaining a change log file. (The database system used by Active Directory does use a transaction log file, but this log is part of the database system, not the replication system.) In a typical **log-based** replication system, each master keeps a log of the updates that it originated. In the log-based system, the goal of each master is to communicate its log to every other replica. After a log has arrived at a replica, the replica applies the log, bringing its state more up-to-date.

The state data includes information that is used to **resolve conflicts** and **avoid sending the full replica on each replication cycle**. Each modification (originating write) operation is assigned a sequence number at its originating domain controller (where that change is made). All replicas maintain information about how up-to-date they are with respect to all other replicas, and values in the directory are tagged with the sequence numbers of their originating write updates. By using this information, the replication source can filter the state changes that it replicates. A state-based approach uses a single mechanism for incremental and full synchronisation.

In general, an Active Directory partition replica maintains all of its objects in a list **ordered** by last modification time. A typical replication request can be satisfied by examining only the last few objects on this list because the replication destination server is aware of how much of its replication source’s list has already been processed.

### Change Notification

Slide Objective

Lead In



Replication within a site occurs as a response to changes. 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, if there are already requests pending, places the request in a queue. Replication occurs one request at a time until all requests in the queue are processed.

When a change occurs on a domain controller, two configurable intervals determine the delay between the following events:

* The change to an attribute and notification of the first partner (initial notification). This interval serves to stagger network traffic caused by 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 notifies all of its replication partners (for that directory partition and within its site) that it has changes. The default value is 15 seconds.
* Notification of the first partner and notification of each subsequent partner (subsequent notification). A domain controller does not notify all of its replication partners at one time. By delaying between notifications, the domain controller spreads out the load of responding to replication requests from its partners. The default delay between notifications is three seconds.

##### Default Notification Values

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 three 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 on a domain controller that is upgraded from Windows 2000 to Windows Server 2003, the non-default value is retained.
* The domain controller has been created from a fresh installation of Windows Server 2003 (not upgraded) and promoted into 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, located in the Partitions container within the configuration directory partition (these attributes do not exist in Windows 2000), as follows:

* 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.

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 earlier.

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

|  |  |
| --- | --- |
|  | HKLMACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters  Replicator notify pause after modify (secs) |

The value for the delay before each subsequent change notification is stored in

|  |  |
| --- | --- |
|  | HKLMACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters  Replicator notify pause between Directory Service and Exchange agents (DSAs) (secs) |

##### 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 freshly installed domain controllers (no upgrade), the registry entries do not exist.)

##### 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, favouring 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 functionality, as follows:

* Windows 2000 operating system and Windows 2000 forest functionality: If a value is set, use this value instead of the default value.
* Windows Server 2003 operating system, Windows 2000 forest functionality: If a value is set, use this value instead of the default value
* Windows Server 2003 operating system, Windows Server 2003 forest functionality: If a value is set, use this value to override the value on the cross-reference object for all directory partitions.

### Active Directory Updates

Slide Objective

Lead In



#### Originating Updates: Initiating Changes

A Lightweight Directory Access Protocol (LDAP) directory server 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.

An LDAP directory server processes each write request as an atomic transaction. Separate LDAP requests are separate write transactions. There are practical limits to the number of values that can be written in one LDAP transaction. It is recommended that updates to the same object be limited to a maximum of 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*. The absolute success or failure of an update applies even for requests, such as add or modify, that might affect several attributes of a single object. In this case, if one attribute update fails, they all fail and the object is not updated.

When an update that originates on one domain controller is replicated to another domain controller, the update on the nonoriginating domain controller is called a replicated update and is distinguishable from an originating update.

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

#### Tracking Updates

Some directory services use timestamps to determine what changes need to be propagated. In these systems, it is important to keep the clocks on all directory servers synchronized. But keeping time closely synchronized in a large network can be difficult because network links fail and clocks drift.

Active Directory replication does not depend on time to determine what changes need to be propagated. It relies instead on the use of **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, decrease in value). The trade-off is that it is meaningless to compare a USN assigned on one domain controller to a USN assigned on a different domain controller. The replication system is designed with this restriction in mind.

Active Directory replication does not use timestamps as the primary mechanism to determine which updates “win” (are preserved) in a conflict resolution. Instead, 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. So if an attribute is updated once on domain controller A and once on domain controller B, the last writer’s update wins. But if the attribute is updated twice on domain controller A and once on domain controller B, the update of domain controller A wins even if the clock of domain controller B is in advance of A. With Active Directory, clock skew can never prevent a value from being overwritten.

**Note:**

Although time synchronization is not required for replication purposes directly Windows 2000 and later do require time synchronization for authentication purposes.

### Resolving Conflicts

Slide Objective

Lead In



Suppose that an attribute of some object is changed on domain controller X. Then before the change on domain controller X has replicated, the same attribute of the same object is changed on domain controller Y. Active Directory must ensure that when replication has occurred, all replicas agree on the value of the updated attribute.

Active Directory ensures agreement by attaching the unique *stamp* to each replicated attribute value (or multivalue) during an originating update. This stamp travels with the value as the value replicates. If the stamp of the value that was replicated is larger than the stamp of the current value, the current value (including the stamp) is replaced; otherwise, the current value (including the stamp) is left alone.

The stamp has the following three components:

* The *version* is a number that is incremented for each originating write. That is, when performing an originating write, the version of the new value is one larger than the version of the value that is being overwritten. If the attribute was never written before, the version that was assigned to its first originating write is 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 a GUID that identifies the domain controller that performed the originating write.

It is possible to see all three components of the stamp in output using **repadmin /showmeta** in Windows 2000 and **repadmin /showobjmeta** in Windows Server 2003. The column labelled “Ver” contains the version, the column labelled “Org. Time/Date” contains the originating time, and the column labelled “Originating DC” contains the originating domain controller (expressed as “site name \server name” rather than GUID).

When stamps are compared, the version is the most significant, followed by the originating time and then the originating domain controller. So 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 domain controller breaks the tie in using the highest GUID.

### Update Sequence Numbers

Slide Objective

Lead In



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) on a domain controller, 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. It is possible to use the Repadmin command-line tool to view the local USN. Type **repadmin /showobjmeta DCLIST <*object\_DN*>** at a command prompt and view the column labelled “Loc. USN” in the output.
* **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. This value can be examined using LDP or ADSIEDIT.

**Note:**

When the forest functionality is Windows Server 2003 or Windows Server 2003 interim, discrete values of linked multivalued attributes can be updated individually 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 travels with the attribute's value as it replicates. Type **repadmin /showobjmeta DCLIST <*object\_DN*>** at a command prompt and view the column labelled “Org.USN” in the output.

### Replication Fundamentals

Slide Objective

Lead In



#### Originating Add

An Add request makes a new object with a unique *objectGuid*. The version number value is set to 1 for all attributes that are populated by the Add request.

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

#### Originating Modify

All Modify requests can be reduced to requests to 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 in the object. If the values are the same, the request to modify that attribute is ignored. If the resulting Modify request does not change any attributes, 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:

1. Sets the *isDeleted* attribute to TRUE.
2. Marks the object as a tombstone, which is an object that has been deleted but not fully removed from the directory.
3. Changes the relative distinguished name to a value that is otherwise impossible (cannot be set by an LDAP application).
4. Strips all attributes that are not needed by Active Directory. A few key attributes, including *objectGuid*, *objectSid*, *distinguishedName*, *nTSecurityDescriptor*, and *usnChanged*, are preserved on the tombstone.
5. Moves the tombstone to the Deleted Objects container, which is a hidden container within the directory partition.

Delivery Tip

Viewing the tombstone is an optional lab exercise.

##### Viewing the tombstone

Object deletions are replicated by replicating tombstones. A tombstone is invisible to normal LDAP searches. (A tombstone is visible to searches that use the special LDAP control 1.2.840.113556.1.4.417.) Object references that formerly referred to the deleted object now refer to the tombstone. Therefore, reading such a reference returns the distinguished name of the tombstone, not the distinguished name of the object prior to the object’s deletion.

|  |  |
| --- | --- |
|  | *For more information, see the following Knowledge Base article: 222560 "Windows 2000 Extended LDAP Controls".* |

##### Garbage Collection: Permanent Removal of Expired Tombstones

Garbage collection is a process that runs on every domain controller to permanently remove expired tombstones from the directory database. Although they represent deleted objects, tombstones take up space in every directory partition replica. Eventually the tombstones themselves must be deleted to keep the directory database from growing without limit. At regular intervals, objects that are no longer needed by the directory service are deleted as "garbage."

One of the functions of the Garbage collection process is to delete tombstones. Garbage collection runs independently on each domain controller. When garbage collection occurs, it finds the set of tombstones whose originating delete occurred more than a tombstone lifetime ago and deletes each tombstone in the set.

Two attributes of the Directory Service object (nTDSService) in the configuration container (CN=Directory Service, CN=Windows NT, CN=Services, CN=Configuration,  
DC=forestRootDomain) control how garbage collection runs and what it removes, as follows:

* Tombstone lifetime determines the number of hours that a deleted object lives as a tombstone in the directory before being collected as garbage, and it is set in the tombstoneLifetime attribute. The default setting is 60 days, and the minimum setting is 2 days.
* Garbage collection interval determines how often a domain controller examines its database for expired tombstones that can be collected, and it is set in the garbageCollPeriod attribute. The default setting is 12 hours, and the minimum setting is one hour.

The maximum garbage collection interval is one-third of the tombstone lifetime (in hours). For example, if the tombstoneLifetime is set to 30 days and garbageCollPeriod to 300 hours, your actual garbage collection period is only 10 days, or 240 hours. ADSIEdit can be used to view or change the default settings for these attributes.

##### Tombstone Lifetime and Active Directory Backup and Restore

Active Directory does not allow restoring from a directory backup that is older than the tombstone lifetime. A restore from backup creates a directory partition replica that has not performed replication since the time of backup (or earlier). If the backup were taken more than a tombstone lifetime before the restore, objects deleted in the meantime would have no tombstones and therefore a new directory partition replica that was created by the restore operation would never receive these deletions. For this reason, a restore procedure will not restore a backup that was taken more than one tombstone lifetime before the time of the restore. It is therefore a recommended best practice to back up Active Directory at least twice during a tombstone lifetime.

Delivery Tip

Organizations such as the military may want to extend the default setting of tombstone lifetime for this reason.

It is important that the tombstone lifetime be substantially longer than the expected replication latency. The default setting of 60 days is generous to accommodate unforeseen circumstances. However, monitoring domain controller operation is essential to ensure that a domain controller does not remain offline without detection.

##### Reanimating the tombstone

In Windows 2000 deleting an object meaning that it is tombstoned the only way to recover it is to restore it from the backup. Windows Server 2003 offers a new feature to reanimate the tombstone effectively allowing for an un-deletion of the object.

The Windows Server 2003 directory database supports an LDAP API that reanimates the tombstone of a single object to avoid the necessity of an offline restore process in the event of administrative error. The API is available for creating applications to restore the attributes that are preserved on tombstones, which include the object SID, GUID, and security descriptor. Only attributes retained on the tombstone are restored; all other data *must* be recreated. Therefore, to restore an entire deleted container or a set of multiple objects, authoritative restore is still the best option.

In order for deleted object restoration to be enabled, at least one domain controller in the domain must be running Windows Server 2003. By default, only domain administrators can restore deleted objects, though this can be delegated to others.

The following limitations apply to restoring deleted objects:

* An object cannot be restored when the tombstone lifetime for the object has expired because when the tombstone lifetime has expired, the object is permanently deleted.
* Objects that exist at the root of the naming context, such as a domain or application partition, cannot be restored.
* Schema objects cannot be restored with this process. However, schema objects could not be deleted in Windows 2000. In Windows Server 2003 there is a feature to mark schema objects as defunct. These defunct objects cannot be restored with reanimation.
* It is possible to restore deleted containers, but the restoration of the deleted objects that were in the container before the deletion is difficult because the tree structure under the container must be manually reconstructed.

|  |  |
| --- | --- |
|  | *For more information, see also…Platform SDK ”Restoring Deleted Objects”* |

### Object Creation

Slide Objective

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



Shows the replication-related data for the user object when it is first created on domain controller DC1. This is known as an originating add.

**Note:**

Many operations including the creation of a user object increment the USN counters by more than one. However for the purposes of this example the slides have simplified this process and show the USNs incrementing by one.

Delivery Tip

It is worth demonstrating this process with replmon after showing the slides. See exercise 3 for instructions.

### Object Replicated

Slide Objective

Lead In



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. This is known as a replicated add.

### Object Modification

Slide Objective

Lead In



This slide illustrates the change in the replicated object on DC2 when someone changes property 2 of the object on that domain controller. By this time, the current USN on DC2 has increased from 1746 to 2001. The update request that changes property 2 increments the current USN to 2002 on DC2. The request also sets the attribute’s originating USN and local USN to 2002 and creates a new stamp for the value. The version number of this properties stamp is incremented by one and is now set to 2. This is known as an originating modify.

### Change Replicated

Slide Objective

Lead In



The changed property is now replicated back to the original domain controller, whose current USN has increased to 5039. The replicated update increments the current USN to 5040. The per-attribute originating USN and stamp are replicated from DC2 to DC1, and the per-attribute local USN and per-object *uSNChanged* values are set to 5040. This is known as a replicated modify.

### Propagation Dampening

Slide Objective

Lead In



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, it might be expected that the same change to be sent along each path to a specific domain controller. Concerns might be raised that a change could replicate in an endless loop. Active Directory prevents these potential problems with multiple paths by using a process known as propagation dampening which uses a number of techniques to prevent these problems, minimizing the amount of network traffic required to support replication.

### High-Watermark

Slide Objective

Lead In



Instructor Note

You might want to explain to non-native English speaking students that high-watermark represents a marking on the side of a ship that indicates its maximum loading.

The high-watermark is a value that the destination domain controller maintains to keep track of the most recent change that it has received from a specific source domain controller for an object in a specific directory partition. The source domain controller uses this value to filter the objects that it considers for replication to the destination.

Each domain controller in the enterprise has a copy of USNs it uses to track the changes it makes. Similarly, each DC in the enterprise keeps a local table where it stores the highest known USN of its replicating partners and their GUIDs. This table is known as the High-Watermark vector, and it is used to detect recent changes on replicating partners.

When requesting changes, the destination domain controller sends its high-watermark value to the source domain controller. The source domain controller uses the information in the high-watermark to reduce the set of objects that it must consider for replication to the destination. No object whose *usnChanged* value is less than or equal to the high-watermark value can hold updates that the destination domain controller has not already received.

The high-watermark serves to decrease the CPU time and number of disk I/O operations that would otherwise be required to send only the changes that the destination domain controller has not yet received.

You can see the high-watermark in the output of the **repadmin /showreps /verbose** command. Look for lines that begin with “USNs:” The high-watermark USN is the number that is followed by “/OU”.

### Up-to-Dateness Vector

Slide Objective

Lead In



In addition to the High-Watermark vector, domain controllers maintain another table which stores originating domain controller GUIDs (database GUIDs), their USNs and a timestamp indicating the last time successful replication took place. This table is referred to as the Up-To-Dateness vector and is Naming Context (NC) specific. This vector is stored in the directory as the "replUpToDateVector" property for the appropriate Naming Context.

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. The source domain controller uses this value to reduce the set of attributes that it sends to the destination domain controller.

When a destination domain controller requests changes for a directory partition, it provides its up-to-dateness vector to the source domain controller. On the basis of this value, the source domain controller can determine that the destination does or does not have an up-to-date value (or multivalue) for an attribute, and it sends updates accordingly. 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.

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.

#### Timestamp on Up-To-Dateness Vector

Windows Server 2003 adds a new field to the UTD in which it records the last time the local DC completed a successful replication cycle with the partner domain controller. The replication cycle may have occurred directly (direct replication partner) or indirectly (transitive replication partner). The timestamp is updated whether or not the local domain controller actually received any changes from the partners.

The timestamp is recorded on all Windows Server 2003 domain controller even if the partner is running Windows 2000. The timestamp is recorded at all domain and forest functional levels.

**Note:**

The timestamps are only updated at the ending of a complete and successful replication cycle. In the case of a long sync or long full sync, the timestamp might not be updated right away although changes are flowing.

It is possible to see the up-to-dateness vector in the output of the **repadmin /showvector** command, adding the /**latency** switch will show the replication latencies within the forest. The replication health of an Active Directory can be quickly assessed because data is recorded on all domain controllers hosting the partition, as it is now possible to identify non-replicating domain controllers. In addition, four new replication events are recorded each time the Microsoft® Active Directory® Knowledge Consistency Checker (KCC) runs. Both errors provide summary information regarding replication failures. The threshold for the first error can be configured in the registry. The default is 24 hours.

|  |  |
| --- | --- |
|  | HKLM\System\CurrentControlSet\Services\NTDS\Parameters  Replicator latency error interval (hours) |

##### Event Messages

There are four new event messages that use the timestamp to identify non-replicating domain controllers and trigger an event specific to a particular problem scenario. Below are the events with a brief description of the problem scenario.

| **Event ID** | **Description** | **Details** |
| --- | --- | --- |
| NTDS Replication 1862 | Non-replicating domain controllers in Other Sites | This is a warning indicating that the local domain controller has not replicated with other domain controller(s) at a different site beyond the latency threshold of 24 hours (configurable). This may be expected if site link schedules prevent replication for more than 24 hours. If that is the case, the warning threshold should be increased to match the site link schedule. |
| NTDS Replication 1863 | Non-replicating domain controllers Local Site and Other Sites | This is the same as the 1862 event (above), but also includes non-replicating domain controllers in the local site. This is a warning indicating that the local domain controller has not replicated with another domain controller in the local site or at a different site beyond the latency threshold of 24 hours (configurable). This may be expected for inter-site partners due to closed schedules. For intra-site partners there are no schedules to observe, so all partner domain controllers should be up to date. If this is recorded, use Repadmin and DCDiag to troubleshoot the affected partner. |
| NTDS Replication 1864 | Summary of domain controllers in the Local Site Not Replicating | This is a summary message of domain controllers from the local site that are no longer replicating. The totals are broken out by length of time. Additionally, the number of domain controllers that have not replicated beyond the tombstone lifetime are reported along with the current tombstone lifetime setting for the forest. |
| NTDS Replication 2042 | Domain Controllers That Have Not Replicated Beyond the Tombstone Lifetime | When a domain controller that has not replicated beyond the tombstone lifetime attempts to replicate, it is blocked and this event is reported. |

### Up-to-Dateness Vector

Slide Objective

The following slides show how ordinary replication takes place.

Lead In



The following example illustrates how ordinary replication takes place. Each domain controller holds an up-to-dateness vector that includes:

* Originating- domain controller -GUID (Database GUID)
* Highest-Originating-USN
* Timestamp

Assume, that only DC1 and DC2 (and maybe DC4) perform originating write operations.

### User added to DC2

Slide Objective

Lead In



A user is added to DC2. This process increments the USN counter for DC2 by one. At this stage there are no changes for DC4.

### User replicated to DC1

Slide Objective

Lead In



DC1 and DC2 initiate replication. DC2 will at this stage send any changes since the last replication took place. DC1 does this by comparing the last USN it knew for DC2 with the current one. The USN counter on DC1 is incremented according to the number of new changes it receives, in this case, one. This value is independent of the one held by DC2. The update is recorded on DC1 as having originated on DC2. There are still no changes on DC4.

### DC4 Initiates Replication with DC1

Slide Objective

Lead In



DC4 and DC1 initiate replication. DC1 will at this stage send any changes since the last replication took place. In order to do this DC 4 sends a GetChange request to DC1. This request includes the following information:

* Naming context (NC) for which changes are requested from.
* If the destination NC replica is partial, its list of partial domain attributes.
* Highest known USN value associated with DC1 for this naming context
* Maximum number of object-update entries that a domain controller can request
* Number of values requested by the domain controller
* Up-to-dateness vector
* Send-Parents (A Boolean value used to determine whether an objects parents need to be sent)

### DC1 replicates new user to DC4

Slide Objective

Lead In



DC1 replies to the GetChanges request with a Getchanges response. This response includes:

* DC1 Server GUID
* DC1 Database GUID
* DC1 schema prefix table
* A sequence of object update entries, each of which includes an Object GUID and a sequence of one or more attribute update entries.
* Last Object USN Changed
* More data (A Boolean value to identify whether there are additional changes over and above the limits set in the request). If this value is TRUE then an additional up-to-dateness vector is sent.

### DC2 replicates new user to DC3

Slide Objective

Lead In



DC3 and DC2 initiate replication. DC2 will at this stage send any changes since the last replication took place. DC3 does this by comparing the last USN it knew for DC2 with the current one. The USN counter on DC3 is incremented according to the number of new changes it receives, in this case, one. This value is independent of the one held by DC2. The update is recorded on DC3 as having originated on DC2. There are no changes on DC4.

### DC4 initiates replication with DC3

Slide Objective

Lead In



DC3 and DC4 initiate replication. DC4 sends a GetChanges request to DC3 in order to receive any changes since the last replication took place.

### DC3 replication reply

Slide Objective

Lead In



DC3 replies with a GetChanges response. It is determined, that DC4 already is up-to-date so the last-object-changed USN and up-to-dateness vector are sent but no data.

### Multimaster Conflict Resolution Policy

Slide Objective

Lead In



As described in earlier in this module, Active Directory resolves attribute value conflicts by comparing versions. However, attribute value conflicts are not the only conflicts that can arise.

Given the semantics of LDAP directories, there are several possible conflicts that are created by multimaster updates:

* 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 value. After resolution: The attribute value at all domain controllers is the value with the larger stamp.
* 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. After resolution: 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 (RDN) 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 RDN values below the same parent object. After resolution: The child object whose naming attribute has the larger 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 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.

**Note:**

In Windows 2000 an RDN conflict was known as a sibling name conflict.

## Lesson 2: Replication Topology

Slide Objective

Lead In



Replication topology is the set of connections by which domain controllers in a forest synchronize the directory partition replicas that they have in common. Replication topology is created on the basis of information stored in Active Directory.

The Knowledge Consistency Checker (KCC) is a built-in process that runs on all domain controllers and creates the replication topology for the forest. By default, the KCC runs at 15‑minute intervals and designates the replication routes between domain controllers on the basis of the most favourable connections that are available at the time. The KCC creates replication connections between domain controllers in the same site automatically. When you have more than one site, you configure links between the sites; the KCC can then create the connections automatically between the sites as well.

### Topology Concepts and Components

Although replication has the effect of synchronizing Active Directory information for an entire forest of domain controllers, the actual process of replication occurs between two domain controllers at a time. Creation of replication topology involves the determination of what domain controller replicates with what other domain controller or domain controllers. When this determination is made for the entire set of domain controllers in a specific site (taking into account that each domain controller must be able to receive changes from all domain controllers in the forest that store the same information), the result is the replication topology for replication within the site. When a forest has domain controllers in more than one site, some of the replication connections between computers must span sites, and a topology for replication between sites is also created.

The total topology is actually composed of several underlying topologies: one for each combination of directory partitions that must be replicated. Domain controllers that store the same domain directory partition must have connections to each other, and all domain controllers must be able to replicate the schema and configuration directory partitions. The schema and configuration directory partitions are replicated over a separate topology; however, where the connections for these directory partitions are identical between domain controllers — for example, two domain controllers store the same domain directory partition — a single connection can be used.

A separate replication topology is also created for application directory partitions. However, in the same manner as schema and configuration directory partitions, application directory partitions can use the same topology as domain directory partitions. When application and domain directory partitions are common to the source and destination domain controllers, the KCC does not create a separate connection for the application directory partition.

The routes for the following combinations of directory partitions are aggregated to arrive at the overall topology:

* Configuration and schema within a site.
* Each domain directory partition within a site.
* Each application directory partition within a site.
* Global catalog read-only, partial domain directory partitions within a site.
* Configuration and schema between sites.
* Each domain directory partition between sites.
* Each application directory partition between sites.
* Global catalog read-only, partial domain directory partitions between sites.

### Topology-related Components

Slide Objective

Lead In



Active Directory uses information stored in the forest-wide configuration directory partition to establish and implement the replication topology. Several configuration objects define the components that are required by replication. The KCC uses these and other objects and their properties to create and manage the connections by which the directory transfers updates and to specify one or more domain controllers from which a particular server requests changes. The domain controllers that replicate directly with each other are called replication partners. Each time the KCC runs these partnerships are added, removed, or modified automatically, as necessary, on the basis of what domain controllers are available and which sites they are in.

The KCC uses the following components to manage replication:

##### Connections

A connection object (class nTDSConnection) defines a one-way, inbound route from one domain controller (the source) to another domain controller (the destination). The KCC uses information in cross-reference objects to create the appropriate connection objects, which enable domain controllers that store the same directory partitions to replicate with each other. The KCC creates connections for every server object in the Sites container that has an NTDS Settings object.

The connection object is a child of the replication destination's NTDS Settings object, and the connection object references the replication source domain controller in the fromServer attribute on the connection object — that is, it represents the inbound half of a connection. The connection object contains a replication schedule and specifies a replication transport. The connection object schedule is derived from the site link schedule for intersite connections. A connection is unidirectional; a bidirectional replication connection is represented as two connection objects under two different NTDS Settings objects.

Connection objects are created in two ways:

* Automatically by the KCC.
* Manually by a directory administrator using Active Directory Sites and Services, ADSI Edit, or scripts.

##### Servers

When installing Active Directory, the installation process creates a server object in the Servers container within the site to which the IP address of the domain controller maps. There is one server object for each domain controller in the site. A server object is distinct from the computer object that represents the computer as a security principal. These objects are in separate directory partitions and have separate GUIDs. The computer object represents the domain controller in the domain directory partition; the server object represents the domain controller in the configuration directory partition. The server object contains a reference to the associated computer object.

##### NTDS Settings Object

The NTDS Settings object (class nTDSDSA) represents an instance of Active Directory on that server and distinguishes a domain controller from a member server in the domain. When Active Directory is removed from a server, its NTDS Settings object is deleted from Active Directory, but its server object remains because the server object might contain objects other than NTDS Settings objects. For a specific server object, the NTDS Settings object contains the individual connection objects that represent the inbound connections from other domain controllers in the forest that are currently available to send changes to this domain controller.

##### Sites

A *site* should represent a region of uniformly good network access, which can be interpreted as being generally equivalent to local area network (LAN) connectivity. LAN connectivity assumes high, inexpensive bandwidth that allows similar and reliable network performance, regardless of which two computers in the site are communicating. This quality of connectivity does not indicate that all servers in the site must be on the same network segment or that hop counts between all servers must be identical. Rather, it can be interpreted as the measure by which you know that if a large amount of data needed to be copied from one server to another, it would not matter to you which servers were involved. If you find that you are concerned about such situations, you might consider creating another site.

Replication between sites occurs according to the schedule; at each site it is possible to use the schedule to determine the most beneficial time for replication to occur on the basis of network traffic and cost. Additionally the frequency with which replication is performed within that period can be specified. A site is the made up of a set of one or more Internet Protocol (IP) subnets.

##### Subnets

Computers on TCP/IP networks are assigned to sites based on their location in a subnet or a set of subnets. Subnets group computers in a way that identifies their physical proximity on the network. Subnet information is used during the process of domain controller location to find a domain controller in the same site as the computer that is logging on. This information also is used during Active Directory replication to determine the best routes between domain controllers.

##### Site Links

For replication to occur between two sites, a link must be established between the sites. Only the first Site link is generated automatically, others can be created in Active Directory Sites and Services. Unless a site link is in place, the KCC cannot create connections automatically between computers in the two sites, and replication between the sites cannot take place. Each site link contains the schedule that determines when replication can occur between the sites that it connects. In addition Site links also specify the cost of the link, which controls the desirability of remote sites as sources of replication information. The Active Directory Sites and Services user interface guarantees that every site is placed in at least one site link. A site link can contain more than two sites, in which case all the sites are treated as equally well connected.

##### Bridgehead Servers

A bridgehead is a point where a connection leaves or enters a site. To communicate across site links, the KCC automatically designates one or more servers, called bridgehead servers, in each site to perform site-to-site replication. At least one bridgehead server per directory partition and per transport (SMTP or IP) replicates changes from one site to a bridgehead server in another site. Replication within the site allows updates to flow between the bridgehead servers and the other domain controllers in the site. Bridgehead servers help to ensure that the data replicated across WAN links is not stale or redundant.

Any server that has a connection object with a "from" server in another site is acting as a destination bridgehead. Any server that is acting as a source for a connection to another site acts as a source bridgehead.

KCC selection of bridgehead servers guarantees that they are capable of replicating all directory partitions, including partial global catalog partitions that are needed in the site. By default, bridgehead servers are selected automatically by the KCC on the domain controller that holds the ISTG role in each site. To identify the domain controllers that can act as bridgehead servers, it is possible to designate preferred bridgehead servers, from which the ISTG selects all bridgehead servers. Alternatively, if the ISTG is not used to generate the intersite topology, then manual intersite connection objects can be created on domain controllers to designate bridgehead servers.

In Windows 2000 forests, a single bridgehead server per directory partition and per transport is designated as the bridgehead server that is responsible for intersite replication of that directory partition. In large hub sites, a single domain controller might not be able to adequately respond to the volume of replication requests from perhaps thousands of branch sites. When the forest has a functional level of Windows Server 2003, the ISTG selects multiple bridgehead servers to balance the replication load.

In both Windows 2000 Server and Windows Server 2003 you can select the set of domain controllers that you want to be responsible for intersite replication. However, the KCC manages failed bridgehead servers differently in Windows Server 2003.

In a Windows 2000 forest or in a Windows Server 2003 forest that has a forest functional level of Windows 2000, the KCC selects a single bridgehead server per directory partition and transport. The selection changes only when the bridgehead server becomes unavailable.

When all domain controllers in the forest have been upgraded to Windows Server 2003 and the forest functional level is raised to Windows Server 2003, the KCC on the ISTG performs an initial random balancing of connections among multiple available bridgehead servers. This load balancing occurs by default, although it can be disabled.

If one or more new domain controllers are added to the hub site, the inbound connections on the existing bridgehead servers are not automatically re-balanced to accommodate the new server or servers. The initial load-balanced connections remain in place until a domain controller becomes unavailable, at which point the KCC randomly replaces the connection on any available bridgehead server.

##### Site Link Bridges

When more than two site links are associated for replication and use the same transport, all of the site links are “bridged” in terms of cost by default, assuming that the site links have common sites. When site links are bridged, they are *transitive*. That is, all site links for a specific transport implicitly belong to a single site link bridge for that transport. So in the common case of a fully routed IP network (in which all sites can communicate with each other by IP), you do not have to configure any site link bridges. If your IP network is not fully routed, you can turn off the transitive site link feature for the IP transport (the **Bridge all site links** option on the **General** tab in the IP transport object property sheet or SMTP transport object property sheet). In this case, all IP site links are considered intransitive, and you configure site link bridges. A site link bridge is the equivalent of a disjoint network; all site links within the bridge can route transitively, but they do not route outside the bridge.

##### Cross-Reference Objects

Cross-reference objects (class crossRef) store the location of directory partitions in CN=partitions,CN=configuration,DC=ForestRootDomain. The contents of the Partitions container are not visible in Active Directory Sites and Services, but can be viewed by using Adsiedit.msc and viewing the Configuration directory partition.

Active Directory replication uses cross-reference objects to locate the domain controllers that store each directory partition. A cross-reference object is created during Active Directory installation to identify each new directory partition that is added to the forest. Cross-reference objects store the identity (nCName) and location (dNSRoot) of each directory partition.

|  |  |
| --- | --- |
|  | *For more information, see also… the Distributed Systems Guide of the Resource Kit* |

### Sites and Replication

Slide Objective

Lead In



#### Site Design with Replication in Mind

When grouping a set of IP subnets into a site, you do so based on the fact that these subnets have high bandwidth, LAN connectivity, possibly involving hops through high-performance routers.

A single domain can span multiple sites, and a single site can contain multiple domains. Domain architecture should be constructed independently of site design — sites exist primarily, but not solely, to assist the domain controller Locator and the replication infrastructure. Although there are no absolute rules for deciding when to place two subnets in the same site, understanding how Active Directory uses site information can help you make an informed decision.

Active Directory uses site information in the following ways:

* When a client requests a connection to a domain controller (for example, when logging on), sites are used to enable the client to connect to a domain controller with good connectivity whenever possible. Fast connections reduce network latency and are usually cheaper.
* When the KCC configures replication connections between domain controllers, it creates more connections between domain controllers in the same site than between domain controllers in different sites. The results are lower replication latency within a site and less replication bandwidth between sites.
* Replication messages between domain controllers within a site are uncompressed, which means that fewer CPU cycles are used on the domain controllers. Replication messages between domain controllers in different sites are compressed, which means that less network bandwidth is used.
* Replication between domain controllers within a site is triggered by update notification messages, which reduces replication latency within a site. Replication between domain controllers in different sites is performed on a schedule, which thereby conserves network bandwidth between sites.

Sites are not tied in any way to the Active Directory namespace that is used by the domain directory partitions. The distinguished name of a domain directory object does not reflect the site or sites in which the object is stored. So the name of a server for a domain moved between sites does not change.

Sites information is stored in the configuration naming context.

|  |  |
| --- | --- |
|  | *For more information on designing a replication topology, see also…*“Designing the Active Directory Structure” in the *Deployment Planning Guide*. |

#### Subnet-to-Site Mapping

Grouping a set of IP subnets into a site should be based on the fact that these subnets have high-bandwidth LAN connectivity, possibly involving hops through high-performance routers.

When creating subnet objects in Active Directory, they can be associated with site objects so that IP addresses can be localized according to sites. Subnet information is used during the process of domain controller location to find a domain controller in the same site as, or the site closest to, the client computer. The Netlogon service on a domain controller is able to identify the site of a client by mapping the client's IP address to a subnet object in Active Directory. Likewise, when a domain controller is installed, its server object is created in the site that contains the subnet that maps to its IP address. Active Directory Sites and Services can be used to define subnets, and then create a site and associate the subnets with the site.

In a default Active Directory installation, there is no default subnet object, so potentially a computer can be in the forest but have an IP subnet that is not contained in any site. For private networks the network addresses can be specified from a range that is defined by the Internet Assigned Numbers Authority (IANA). By definition, that range covers all of the subnets for the organization. However, where several class B or class C addresses are assigned, there would necessarily be multiple subnet objects that all mapped to the same default site.

To accommodate this situation, use the following subnets:

* For class B addresses, subnet 128.0.0.0/2 covers all class B addresses.
* For class C addresses, subnet 192.0.0.0/3 covers all class C addresses.

|  |  |
| --- | --- |
|  | *For more information, see also…*"Introduction to TCP/IP" in the Networking Guide of the Resource Kit |

#### Default Site

When installing Active Directory on the first domain controller in the forest, a site object named Default-First-Site-Name is created in the Sites container in Active Directory. The server object for the first domain controller is created in this site. When installing Active Directory on subsequent servers, if additional sites have not been defined in Active Directory, the new domain controller's server object is created in the site of the source domain controller. If additional sites have been defined in Active Directory and the IP address of the installation computer matches an existing subnet in a defined site, the domain controller is added to that site.

#### Default Site Link

When installing Active Directory on the first domain controller in the forest, an object named DEFAULTIPSITELINK is created in the Sites container in Active Directory (the IP container within the Inter-Site Transports container). This site link contains only one site, Default-First-Site-Name.

### Replication Transports

Slide Objective

Lead In



Replication transports provide the wire protocols that are required for data transfer. There are three levels of connectivity for replication of Active Directory information:

* Uniform high-speed, synchronous remote procedure call (RPC) over IP within a site.
* Point-to-point, synchronous, low-speed RPC over IP between sites.
* Low-speed, asynchronous SMTP between sites.

The following rules apply to the replication transports:

* Replication within a site always uses RPC over IP.
* Replication between sites can use either RPC over IP or SMTP over IP.
* Replication between sites over SMTP is supported for only domain controllers of different domains. Domain controllers of the same domain must replicate by using the RPC over IP transport. Therefore, replication between sites over SMTP is supported for only schema, configuration, and Global Catalog replication, which means that domains can span sites only when point-to-point, synchronous RPC is available between sites.

The Inter-Site Transports container provides the means for mapping site links to the transport that the link uses. When creating a site link object, create it in either the IP container (which associates the site link with the RPC over IP transport) or the SMTP container (which associates the site link with the SMTP transport).

|  |  |
| --- | --- |
|  | Closer Look: Replication TransportsSynchronousversus Asynchronous Communication The RPC inter- and intrasite transport (RCP over IP within sites and between sites) and the SMTP intersite transport (SMTP over IP between sites only) correspond to synchronous and asynchronous communication methods, respectively. Synchronous communication favours fast, available connections, while asynchronous communication is better suited for slow or intermittent connections.  The IP transport (RPC over IP) provides synchronous replication. In the context of Active Directory replication, synchronous communication implies that after the destination domain controller sends the request for data, it waits for the source domain controller to receive the request, construct the reply, and send the reply before it requests changes from any other domain controllers; that is, inbound replication is sequential. Thus in synchronous transmission, the reply is received within a short time. The IP transport is appropriate for linking sites in fully routed networks.  The SMTP transport (SMTP over IP) provides asynchronous replication. In asynchronous replication, the destination domain controller does not wait for the reply and it can have multiple asynchronous requests outstanding at any particular time. Thus in asynchronous transmission, the reply is not necessarily received within a short time. Asynchronous transport is appropriate for linking sites in networks that are not fully routed and have particularly slow WAN links. Dynamic Replication Ports 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. Transports for Replication Between Sites Active Directory supports two default transports for replication between sites:   * RPC over TCP/IP (referred to as “IP” in administrative tools), which enables low-speed, point-to-point, synchronous compressed replication between all directory partitions. * SMTP, which enables low-speed, asynchronous replication between the schema, configuration, and Global Catalog directory partitions, but not between domain directory partitions.   When sites are on opposite ends of a WAN link (or the Internet), it is not always desirable — or even possible — to perform synchronous, RPC-based directory replication. In some cases, the only method of communication between two sites is e‑mail. To support such configurations, replication must be possible across asynchronous, store-and-forward transports such as SMTP.  For intersite replication, SMTP replication substitutes mail messaging for the RPC transport. The message syntax is the same as for RPC-based replication. There is no change notification for SMTP-based replication, and scheduling information on the site link object is used as follows:   * By default, SMTP replication ignores the Replication Available and Replication Not Available settings on the site link schedule in Active Directory Sites and Services (the information that indicates when these sites are connected). * SMTP replication uses the replication interval to indicate how often the server requests changes. The interval (Replicate every \_\_\_\_ minutes) is set in 15-minute intervals on the General tab in site link Properties in Active Directory Sites and Services.   These settings combine to form the replication schedule on the connection object. The underlying SMTP messaging system is responsible for message routing between SMTP servers. Comparison of SMTP and RPC Replication The following characteristics apply to both SMTP and RPC with respect to Active Directory replication:   * For replication between sites, data that is replicated through both transports is compressed. * Active Directory can respond with only a fixed (maximum) number of changes per change request, on the basis of the size of the replication packet. The size of the replication packet is configurable. (For information about configuring the replication packet size, see "Replication Packet Size" later in this chapter.) * Active Directory can have only a single change request outstanding for a specific directory partition to a specific replication partner. * The response data (changes) are transported in one or many <a ID="wPopUp" HREF="HELP=glossary.hlp TOPIC=gls\_frames">frames</a>, based on the total number of changed or new values. * TCP transports the data portion by using the same algorithm for both SMTP and RPC. * If transmission of the data portion fails, complete retransmission is necessary. * If bandwidth is limited, the same TCP retransmission characteristics apply. (RPC time-out is much longer than TCP time-out.)   Point-to-point synchronous RPC replication is available between sites to allow the flexibility of having domains that span multiple sites. RPC is best used between sites that are connected by WAN links because it involves lower latency. SMTP is best used between sites where RPC over IP is not possible. For example, SMTP can be used by organizations that have a network backbone that is not based on TCP/IP, such as those that use an X.400 backbone.  Active Directory replication uses both transports to implement a request-response mechanism. Active Directory issues requests for changes and replies to requests for changes. RPC maps these requests into RPC requests and RPC replies. SMTP, on the other hand, actually uses long-lived TCP connections (or X.400-based message transfer agents in non-TCP/IP networks) to deliver streams of mail in each direction. Thus, RPC transport expects a response to any request immediately and can have a maximum of one active inbound RPC connection to a directory partition replica at a time. The SMTP transport expects much longer delays between a request and a response. As a result, multiple inbound SMTP connections to a directory partition replica can be active at the same time, provided the requests are all for a different source domain controller or directory partition. Replication Packet Size By default, packet sizes are computed on the basis of memory size unless you have more than 1 gigabyte (GB) or less than 100 megabytes (MB) of memory. You can override these memory-based values in the registry.  To adjust the default size of the packets that transport Active Directory replication data, you can modify or add entries to the following registry path with the REG\_DWORD data type:  HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\NTDS\Parameters.  These entries determine the maximum number of objects per packet and maximum size of the packets.   * For RPC replication within a site:   Replicator intra site packet size (objects)  Range: >=2  Replicator intra site packet size (bytes)  Range: >=10 KB   * For RPC replication between sites:   Replicator inter site packet size (objects)  Range: >=2  Replicator inter site packet size (bytes)  Range: >=10 KB   * For SMTP replication between sites:   Replicator async inter site packet size (objects)  Range: >=2  Replicator async inter site packet size (bytes)  Range: >=10 KB  If the preceding registry entries are not set, the system limits the packet size as follows:   * The packet size in bytes is 1/100th the size of RAM with a minimum of 1 MB and a maximum of 10 MB. * The packet size in objects is 1/1,000,000th the size of RAM, with a minimum of 100 objects and a maximum of 1,000 objects.   There is one exception: the value of the Replicator async inter site packet size (bytes) entry is always 1 MB. Many mail systems limit the amount of data that can be sent in a mail message (2 MB to 4 MB is common), although most Windows-based mail systems can handle large 10 MB mail messages. |

### Global Catalog Replication

Slide Objective

This is an animated slide first showing replication of full NCs schema and config then showing a series of replications from full to partial NC and finally from partial to partial NC.



A global catalog server is a domain controller that stores information about all objects in the forest so that applications can search Active Directory without having to be referred to specific domain controllers that store the requested data. Like all domain controllers, a global catalog server stores full, writable replicas of the schema and configuration directory partitions, and a full, writable replica of the domain directory partition for which the domain controller is authoritative. In addition, a global catalog server stores partial, read-only replicas of all other domain directory partitions in the forest. When an attributeSchema object has the isMemberOfPartialAttributeSet attribute set to TRUE, the attribute is replicated to all global catalog servers in the forest, in addition to the corresponding directory partition replicas on all authoritative domain controllers.

Global catalog servers can speed Active Directory searches and facilitate logons (a requirement for Active Directory) on the one hand, and can create increased replication traffic on the other. Whether you enable a global catalog server in a site depends on the needs of the users and applications and on the speed and availability of connections to other sites.

The first domain controller in the forest is designated automatically as a global catalog server. When additional sites are created, use Active Directory Sites and Services to enable a global catalog server for that site.

Before a domain controller advertises itself as a global catalog server in DNS, the entire global catalog must be replicated to the server. This process involves replication of a partial replica of every domain object in the forest for every domain other than the domain for which the new global catalog server is authoritative. How long this process takes depends on how many domains the forest contains and the relative locations of domain controllers. If there are multiple domains and source domain controllers are located only in distant sites, the process will take longer than if all domains are in the same site or in only a few sites. When replication must occur between sites to create the global catalog, the site link schedule determines when replication can occur.

After a domain controller has been designated as a global catalog server, the KCC updates the topology and replication of the global catalog partial directory partitions to the new global catalog server proceeds after the KCC performs a topology check. When the KCC runs, it checks to see whether the global catalog option is selected for any domain controllers, and creates the replication topology accordingly. The KCC configures the newly selected global catalog server to be the destination server for a read-only replica of each domain directory partition in the forest that the server does not already hold as a writeable copy. The KCC on the global catalog server must be able to reach a server that will be the source of each read-only directory partition.

When the KCC locates an available source domain controller, it creates an inbound connection on the new global catalog server and replication of that read-only partition takes place. If the source is within the site, replication begins immediately. If the source is in a different site, replication begins at the next scheduled replication window. Replication of all objects in the partial directory partition must complete successfully before the directory partition is considered to be present on the global catalog server.(if <sp3, only domains within local site. For >=sp3, All Domains in the forest. )

When all directory partitions are present, the domain controller sets its rootDSE attribute isGlobalCatalogReady to TRUE and the Net Logon service on the domain controller registers global-catalog-specific service resource records (SRV) in DNS. At this point, the global catalog is considered to be available.

Global catalog servers request updates from a source domain controller for each domain directory partition in the forest (they generate inbound connection objects from those domain controllers). The source domain controller for replication of a given directory partition to a global catalog server can be either a normal domain controller or another global catalog server. As is true for all domain controllers, a global catalog server uses a single topology to replicate the schema and configuration directory partitions, and it uses a separate topology for each domain directory partition.

##### Replication of Changes to the Global Catalog Partial Attribute Set

The default set of attributes that are replicated to the global catalog are identified by the schema. These attributes are referred to as the "partial attribute set" because they provide a replica of every object in the directory, but only those attributes that are most likely to be used for searches. To add an attribute to the partial attribute set, it can be marked by editing the isMemberOfPartialAttributeSet value on the respective attributeSchema object. If the value is set to TRUE, the attribute is replicated to the global catalog. When a schema change affects the set of attributes that are marked for inclusion in the global catalog (an attribute is added to the partial attribute set), replication of the change occurs differently on global catalog servers running Windows 2000 and those running Windows Server 2003, as follows:

* Both servers running Windows 2000: The global catalog server initiates a full synchronization of all partial, read-only domain directory partition replicas in order to become up-to-date with the extended replica image on other domain controllers. If the partial directory partition replica can be synchronized over an RPC connection, the domain controller attempts a full synchronization over the RPC connection before it uses an SMTP connection; if full synchronization is completed, the up-to-dateness vector that it creates optimizes later full synchronization on other connections.
* Both servers running Windows Server 2003: Only the changed attributes are replicated to global catalog servers that are running Windows Server 2003. There is no replication impact.
* One global catalog server running Windows 2000, the other running Windows Server 2003: If a global catalog server that is running Windows Server 2003 replicates the change to a global catalog server that is running Windows 2000, the Windows Server 2003 reverts to Windows 2000 behavior. Although interruption of service does not occur, this replication causes higher bandwidth consumption than is required for usual day-to-day replication. The resulting bandwidth consumption for each global catalog server is equivalent to that caused by promoting a regular domain controller to the role of global catalog server.

The deletion operation does not involve replication, but is handled locally. If you set the isMemberOfPartialAttributeSet value to FALSE in the schema, the attribute is removed from the global catalog immediately after the next replication cycle. This behavior is the same on global catalog servers running Windows Server 2003 and Windows 2000.**(but we must wait for the completion of the promotion**.)

### KCC and Topology Generation

Slide Objective

Lead In



The KCC is a built‑in process that runs on all domain controllers. It is a dynamic link library (DLL) that modifies the replication topology in response to system-wide changes. The KCC generates and maintains the replication topology for replication within and between sites. A component of the KCC called the Inter-site Topology Generator (ISTG) is responsible for building the inter-site connections.

The KCC has two major functions:

* Configures replication connection objects between domain controllers. Each connection object defines incoming replication from a replication partner. Within a site, each KCC generates its own connections. For replication between sites, a single ISTG per site generates all connections between sites. Connection objects can be viewed in Active Directory Sites and Services or by using an LDAP tool to view a server’s NTDS Settings object in the Configuration container.
* Converts the connection objects that represent inbound replication to the local domain controller into the replication links that are actually used by the replication engine.

By default, the KCC reviews and makes modifications to the Active Directory replication topology every 15 minutes to ensure propagation of data, either directly or transitively, by creating and deleting connection objects as needed. The KCC recognizes changes that occur in the environment and ensures that domain controllers are not orphaned in the replication topology.

#### Tools That Communicate with the KCC

The automatically generated connection objects and the UI component for managing the KCC are visible in the Active Directory Sites and Services Microsoft Management Console (MMC) snap-in. Most replication tasks that affect the KCC can be managed by using Active Directory Sites and Services, Replmon, and Repadmin.

##### Active Directory Sites and Services

Active Directory Sites and Services is the primary administrative tool that is used to manage replication topology. This tool can be used to create connection objects and site links that the KCC uses to implement replication. Replication within a site is completely automatic and usually requires no administrative intervention. However, an administrator can tune and refine the behaviour of the ISTG by defining preferred bridgeheads, site links, schedules and site link bridging options. In larger environments, such administrative tasks are more of a requirement.

##### Event Viewer

In Windows Server 2003, significant improvements have been made to Directory Services event logging, including KCC event messages such as those in the table below. An example is the Windows 2000 event 1311 message logged due to insufficient connectivity in the domain or forest. In Windows Server 2003, the affected partition is now listed to help determine the root cause. In addition, new messages specify which Active Directory site is affected by connectivity issues as well as specifying sites that are not included in site links.

| **Event ID** | **Description** |
| --- | --- |
| 1311 | Lists Partition that is unable to Replicate from this site |
| 1789 | Site Uncovered by any Site Link |
| 1865 | Lists the Sites unable to complete the spanning tree |
| 1925 | Unable to Build Replication Link |
| 1308 | Routing Around domain controller failure |
| 1566 | All domain controllers in given site are unable to replicate |
| 1567 | Preferred Bridgehead server defined but unable to replicate all partitions in the site |
| 1864 | Summary of Non-Replicating domain controllers |

##### Logging Levels

The KCC, like all subsystems in Active Directory, has a variable event logging level. By default, only the most important events are logged. You can increase the level of detail in the event log by modifying the value in the **Replication Events** entry in the following key.

|  |  |
| --- | --- |
|  | HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Diagnostics |

Increasing the level of detail can be used for better understanding the behavior of the KCC in different situations. However, a logging level value greater than 2 generally results in excessive logging that degrades the performance of the component. Increasing the logging level can be useful for troubleshooting problems, but it is not recommended for normal operation.

#### Objects Required by the KCC for Building Topology

When the KCC builds the topology, it must determine which servers are present in each site in order to construct an efficient topology. The following objects provide the information required by the KCC to create the topology:

* Server object: All domain controllers are identified as server objects in the configuration directory partition, broken down by site.
* The NTDS Settings object: Each server object that represents a domain controller has a child NTDS Settings object. The NTDS Settings object must be present for the server to be considered as part of the replication topology.

The presence of these objects also determines the site in which the domain controller is to be located. For example, the distinguished name of the NTDS Settings object contains the site to which that domain controller belongs. If the server is physically located in one site but is configured for another site in Active Directory, the KCC uses the information in Active Directory to construct the topology. Therefore, the improper configuration of servers in sites can affect network bandwidth.

#### Configuring the KCC

The KCC evaluates the replication topology at specified intervals, which can be modified.

By default, the KCC runs its first replication topology check five minutes after the domain controller starts. Changing the following registry entry can modify this interval.

|  |  |
| --- | --- |
|  | HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters  Repl topology update delay (secs)  Value: Number of seconds to wait between the time Active Directory starts and the KCC runs for the first time.  Default: 300 seconds (5 minutes)  Data type: REG\_DWORD |

By default, as long as services are running, the KCC checks the topology every 15 minutes and makes changes as necessary. The administrator can modify the interval at which the KCC performs this review by changing the following entry.

|  |  |
| --- | --- |
|  | HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters  Repl topology update period  Value: Number of seconds between KCC topology updates  Default: 900 seconds (15 minutes)  Data type: REG\_DWORD |

### *Inter-Site Topology Generator(latest)*

Slide Objective

Lead In



Every site has one domain controller that is automatically selected to act as the intersite topology generator (ISTG) for the site. The KCC on the ISTG is responsible for bridgehead server selection: creating the inbound connections on all domain controllers in its site that require replication with domain controllers in other sites. The sum of these connections for all sites in the forest forms the intersite replication topology.

A fundamental concept in the generation of the topology within a site is that each server does its part to create a sitewide topology. In a similar manner, the generation of the topology between sites depends on each site doing its part to create a forest-wide topology between sites.

It is not possible to manually change the ISTG role in a site. The owner of the ISTG role is communicated through normal Active Directory replication. Initially, the first domain controller in the site is the ISTG. The role does not change unless:

* The current ISTG becomes unavailable.
* All domain controllers in the site are running Windows 2000 and one of them is upgraded to Windows Server 2003.

If at least one domain controller in a site is running Windows Server 2003, the ISTG role is assumed by a domain controller that is running Windows Server 2003.

#### ISTG Role Owner Viability

The ISTG communicates its role ownership to other domain controllers in the site by writing the interSiteTopologyGenerator attribute on the NTDS Site Settings object for the site, which is replicated to all domain controllers in the forest. The viability of the current ISTG is assessed by all other domain controllers in the site. The need for a new ISTG in a site is established differently, depending on the forest functional level that is in effect.

* Windows 2000 functional level: At 30 minute intervals, the current ISTG notifies every other domain controller of its existence and availability by writing the interSiteTopologyGenerator attribute on the NTDS Site Settings object for the site. The change replicates to every domain controller in the forest. The KCC on each domain controller monitors this attribute for its site to verify that it has been written. If a period of 60 minutes elapses without a modification to the attribute, a new ISTG declares itself.
* Windows Server 2003 or Windows Server 2003 interim functional level: Each domain controller maintains a table of the times that it was last contacted by its replication partners, including both direct and indirect partners (that is, every domain controller that replicates a directory partition that is stored by this domain controller). Because all domain controllers store the schema and configuration directory partitions, every domain controller is guaranteed to have the ISTG for its site among the domain controllers in its last-contacted table. This table, which is maintained as part of the server’s replication metadata, eliminates the need to receive periodic replication of the updated interSiteTopologyGenerator attribute from the current ISTG. When the table indicates that the current ISTG has not been heard from in the last 120 minutes, a new ISTG declares itself.

The Windows Server 2003 method eliminates the network traffic that is generated by periodically replicating the interSiteTopologyGenerator attribute update to every domain controller in the forest.

#### ISTG Selection

The process for selecting a new ISTG role owner is the same for all forest functional levels, but does consider the operating system of the domain controllers to establish eligibility. When a new ISTG is required, each domain controller computes a list of domain controllers in the site that are eligible to be the ISTG. Eligibility is established as follows:

* If no domain controllers in the site are running Windows Server 2003, all domain controllers that are running Windows 2000 Server are eligible.
* If at least one domain controller in the site is running Windows Server 2003, all domain controllers that are running Windows Server 2003 are eligible.

All domain controllers in the site arrive at the same list, and they sort the list by GUID. The domain controller that is first in the sorted list declares itself the new ISTG by writing the interSiteTopologyGenerator attribute on the NTDS Site Settings object.

### KCC Scalability

Slide Objective

Lead In



#### Windows 2000 Limitations

The Windows 2000 KCC has a scalability limitation that results in high CPU utilization during the calculation of the topology. The complexity of the topology increases with the addition of sites or domains and the enabling of site link bridging. Increasing the number of sites causes an exponential increase in runtime. When the number of domains and sites within a single forest reaches a threshold, the KCC is no longer able to keep up (i.e., KCC run time execution exceeds its run time frequency so that it runs continuously and prevents other process from completing).

If D is the number of domains in the network, S is the number of sites in a network, and

(1 + D) \* S^2 <= 100,000

then the KCC should be able to perform adequately.

#### Windows Server 2003 Performance Enhancements

A number of performance enhancements have been incorporated into the Windows Server 2003 KCC. These include improvements to the Windows 2000 spanning tree algorithm (used for backward compatibility) and a new spanning tree algorithm that is significantly more efficient than the previous version.

##### Windows Server 2003 DCs in Windows 2000 Native Forest Mode

The Windows Server 2003 KCC will operate differently depending on the forest functional level (or *mode*). In order to gain the full benefits of the improved KCC algorithms and scalability, the forest functional level must be set to Windows Server 2003 interim or Windows Server 2003.

If the forest functional level is set to Window 2000, the KCC operates much like the Windows 2000 KCC. For example, it uses the same algorithm to calculate the spanning tree. This is done to ensure backward compatibility with existing Windows 2000 domain controllers.

Even with the forest functional level set at Windows 2000, refinements in Windows Server 2003 allow a small increase in KCC performance without leveraging the new enhanced algorithm. To realize these initial benefits, all of the ISTGs in the forest (one per site) must be running on a Windows Server 2003 server. This means that at least one Windows Server 2003 domain controller must be installed at all sites. Once a Windows Server 2003 domain controller is added to a site, it will automatically assume the role of the ISTG, and the initial performance gains will be realized. The performance gains are relatively small. In a large environment, setting the forest functional level to Windows Server 2003 may be required in order to effectively deal with the expensive calculations associated with more sites and domains. When this change is made if D is the number of domains in the network, S is the number of sites in a network, and

(1 + D) \* S <= 100,000

then the KCC should be able to perform adequately.

##### Bridgehead load distributing

A common problem in branch office scenarios is that one bridgehead server in the hub site is selected for all the outbound replication to the branch sites. Other domain controllers in the hub site are under-utilized while the one bridgehead server may become overloaded servicing downstream replication.

In Windows 2000, this scenario is often addressed by one of two methods:

* Placing each hub domain controller in its own Active Directory site, and then balancing site links among all available hub domain controllers.
* Disabling the ISTG function and creating the connection links manually. A set of scripts (collectively referred to as MKDSX) to automate this process is included in the *Active Directory Branch Office Deployment Guide*.

The Windows Server 2003 ISTG uses a “random selection” process when selecting the bridgehead for a site. The result is that the load for downstream partners is more evenly distributed among the domain controllers in the hub site. The random selection process only occurs when a new connection link is added. If additional domain controllers are added at the hub site, the KCC will not redistribute the load. This feature can be disabled by setting the value of the “options” attribute of the “NTDS Site Settings” object for the site to 512.

###### Enhanced Event Messages for 1311 errors

A common issue with Windows 2000 Active Directory is a 1311 event message from the KCC indicating that there is not enough published connectivity to create a spanning tree. In essence, an error condition exists (either configuration or replication), which prevents the successful end-to-end replication of all partitions. The 1311 event message gives insufficient detail to troubleshoot the issue.

Windows Server 2003 includes a number of enhancements to the 1311 event messages as well as a number of new messages that clearly define the issue and provide the required detail to properly troubleshoot these problems. Because it is the job of the KCC on the Inter-site Topology Generator to look for and report these conditions, the new event messages are exposed as soon as a Windows Server 2003 domain controller is added to a site, and assumes the ISTG role.

##### Windows Server 2003 Forest Functional Level

Once all of the Windows 2000 domain controllers have been upgraded or replaced with Windows Server 2003 domain controllers, the forest functional level can be increased to Windows Server 2003. This enables all of the new performance enhancements in the Windows Server 2003 KCC. The changes introduced at this level are:

* Windows Server 2003 KCC spanning tree algorithm enabled, full performance benefits realized.
* KCC can be used for topology generations in deployments up to 3000 sites.
* KCC no longer pulls cost matrix data from the inter-site messenger service (ISM).

The new algorithm replaces the function of the ISM to evaluate the least cost route with a much more efficient algorithm. The old ISM algorithm was much more expensive, but it exhaustively tried every possible combination. This allowed it to perform better backtracking when a route was found undesirable. The old algorithm could almost always find the best path based on cost and schedule. The new algorithm is primarily based on cost, and may not select the same route as the old algorithm when schedule is a secondary factor. The ISM still exists in Windows Server 2003 and still uses the expensive exhaustive algorithm. However, it is not used by the KCC if the forest functional level is set to Windows Server 2003. If the forest functional level is set to Windows 2000, the ISM *is* used by the KCC, as well as by Netlogon, DFS, SMTP replication, no-Global Catalog Logon, and for calculating cost to alternate-site links when an in-site DFS target is not available. Even though the ISM is not used by the KCC, it will likely be the scalability bottleneck for number of Active Directory sites and domains. Improvements over Windows 2000 have been made to ISM to run more efficiently.

* Staggered schedule creation can be automated with the Active Directory Load Balancing tool.
* Staggered replication start times can be enabled so all downstream partners do not start replicating as soon as the replication window opens.

### Automatic Replication Topology Generation Within a Site

Slide Objective

This is an animated slide

Lead In



* In the simple case of one domain with one domain controller there is no need for replication.
* When DC2 is added a pair of connection objects to replicate all three naming contexts are created between it and DC1.
* When DC3 is added two more pairs of connection objects are added to the topology
* When DC4 is added an additional pair of objects are created between it and DC2 and DC3. This results in a redundant set of connection objects between DC2 and DC3 and as such the KCC will eventually remove them

#### Simplified Ring Topology Generation

An overly simplified process for creating the topology for replication within a site begins as follows:

The KCC generates a list of all servers in the site that hold that directory partition.

These servers are connected in a ring ordered by database GUID number.

For each neighboring server in the ring from which the current domain controller is to replicate, the KCC creates a connection object if one does not already exist.

This simple approach guarantees a topology that tolerates a single point of failure. If a domain controller is not available, it is not included in the ring that is generated by the list of servers because its NTDS Settings object is not available. However, this topology, with no other adjustments, accommodates only seven servers. Beyond this number, the ring would require more than three hops for some servers.

The simplest case scenario — seven or fewer domain controllers, all in the same domain and site — would result in the topology shown in the slide. Even if one or all of these domain controllers were Global Catalog servers, when the KCC runs on those particular computers, no extra connections would be necessary. The only directory partitions to replicate are a single domain directory partition, the schema directory partition, and the configuration directory partition. Those topologies are generated first, and at that point, sufficient connections to replicate each directory partition have already been created.

### Two Domains within a Site

Slide Objective

This is an animated slide

Lead In



Because a ring topology is created for each directory partition, the topology might look different if domain controllers from a second domain were present in the site. The slide illustrates the topology for domain controllers from two domains in the same site with no Global Catalog servers defined in the site.

* If an additional domain is added to the configuration then sets of connection objects to replicate the configuration and schema naming contexts to the new domain controller. The domain naming context is not replicated to the new domain controller.
* When DC2 for sales.microsoft.com is added, connection objects between it and DC1 are created to replicate all the naming contexts while sets of connection objects to replicate the configuration and schema naming contexts begin to appear.

### Expanded Ring Topology

Slide Objective

This is an animated slide

Note that more than one optimising connection is likely to be added in this scenario.



When the number of servers grows, the KCC estimates the number of connections that are needed so that if a change occurs at any one domain controller, there are as many replication partners as needed. These optimizing connections are created at random and are not necessarily created on every domain controller. Based on the number of domain controllers in the site, n, the KCC computes a number f(n), which is the number of random shortcut connections which should be added to each domain controller. If f(n) random shortcut connections are added to each domain controller, the number of hops from any domain controller to any other domain controller should be roughly three. It does not guarantee a maximum number of hops from one domain controller to another. It is unlikely that it would be necessary to go 5 hops to get from one domain controller to another within a site.

In the slide, there is no Global Catalog server in the site, all domain controllers are in the same domain, but enough servers have been added to require optimising connections. Although they are located in the same domain and site, DC2 and DC8 are several hops away from each other. The optimising connections for the domain, schema, and configuration directory partitions that might be created from DC2 to DC8 are depicted as a single straight line in the diagram for readability, but in reality, these partitions are replicated separately as shown between the neighboring replication partners. There would also be more optimizing connections than the one shown.

### Urgent Replication

Slide Objective

Lead In



Certain important security and time sensitive changes to objects and settings trigger replication immediately, overriding existing change notification and schedule settings. Urgent replication is implemented by immediately notifying replication partners over RPC 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 change notification has been enabled between sites, these triggering events also replicate immediately between sites.

The test for whether to use urgent replication is based on whether any aspect of the system experiences negative side effects due to replication latency. For example, long replication latencies for the rid manager may cause objects with duplicate SIDS to be created. Immediate notification between Active Directory based domain controllers in the same site is prompted by the following:

* 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 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.

In a mixed Windows NT 4.0 and Active Directory domain there are a number of additional events that trigger urgent replication. These are listed in the article referenced below.

|  |  |
| --- | --- |
|  | *For more information, see the following Knowledge Base article:* 232690 *"*Urgent Replication Triggers in Windows 2000 *".* |

#### 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.

Account lockouts are urgently replicated to the primary domain controller (PDC) emulator and are 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.

### Password Replication

Slide Objective

Lead In



Changes to account passwords can be made at any domain controller because all full replicas of a given domain are writable. This differs from Windows NT 4.0 and earlier versions, in which password changes were made at the PDC for the domain. This is the only writable replica of the Security Account Manager in Windows NT 4.0. This can lead to unexpected behavior when a password is changed by a user at domain controller "A" who then attempts to log on with authentication by domain controller "B." If the password has not been replicated from "A" to "B," the logon attempt does not succeed. In Windows NT 4.0, if authentication does not succeed at the BDC, the authentication is remoted to the PDC. Active Directory exhibits behavior as follows:

* A password change by a Directory Service-aware client at a domain controller is "pushed" by that domain controller to the PDC Microsoft® Active Directory® Flexible Single Master Operations (FSMO) role owner on a best-effort basis. This push of the password to the PDC can be disabled on WAN links with the following registry key:

|  |  |
| --- | --- |
|  | HKLM\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters  Registry value : AvoidPdcOnWan  Registry type : REG\_DWORD  Registry value data : 0 (or value not present) or 1  FALSE = 0 or value not present (to disable)  TRUE = 1 (to enable)  Default : (value is not present)  Platform : Only Windows 2000 Domain Controllers |

If the AvoidPdcOnWan value is set to TRUE and the PDC FSMO is located at another site, the password change is not sent immediately to the PDC. However, it is notified of the change through normal Active Directory replication, which in turn replicates it to down-level domain controllers (if the domain is in mixed mode). If the PDC FSMO is at the same site, the AvoidPdcOnWan value is disregarded and the password change is immediately communicated to the PDC. The above registry value can be used by the administrator to control when the PDC is contacted, which can help reduce communication costs between sites.

* The password change is propagated to other domain controllers in the domain using normal replication values.
* When authentication does not succeed at a domain controller other than the PDC FSMO role owner, the request is retried at the PDC FSMO role owner. If the AvoidPdcOnWan value is set to TRUE and the PDC FSMO role owner is located at another site, the domain controller does not try to authenticate a client against password information stored on the PDC FSMO. Note, however, that this results in denying access to the client.
* Down-level clients attempt to contact the PDC to make a password change as they do in Windows NT 4.0.

The “User Must Change Password at Next Logon” setting sets the PwdLastSet attribute to zero. This indicates that the password has expired and the user should be prompted to change the password before they can log on. After a user changes his or her password, this attribute stores the UTC time when the password was changed. Changes to this value are not replicated urgently or pushed to the primary domain controller. This can cause potential logon failures or account lockouts if an administrator changes an account password at the same time as setting this option and does not perform the operations at the PDC emulator. This is because the new password could arrive at the PDC before the password expiry option does.

#### Change Notification between Sites

By default, changes are replicated between sites according to a schedule and not according to when changes occur. For this reason, the greatest replication latency across the forest is the sum of the greatest replication latencies along the single longest replication path of any directory partition. For special circumstances, it is possible to configure change notifications on connections between sites. By modifying the site link object, change notifications can be enabled between sites for all connections that occur over that link. Use ADSI Edit to enable change notification between sites.

To enable change notification between sites:

1. In ADSI Edit, expand the Configuration container.
2. Navigate to the Inter-Site Transports container, and select CN=IP. (Change notification for SMTP links cannot be enabled).
3. Right-click the site link object for the sites for which you want to enable change notification, and then click Properties.
4. In the Select a property to view box, select options.
5. In the Edit Attribute box, if the Value(s) box shows <not set>, type 1 in the Edit Attribute box.

If the Value(s) box contains a value, you must derive the new value by using a Boolean BITWISE-OR calculation on the old value, as follows: old\_value BITWISE-OR 1.

For example, if the value in the Value(s) box is 2, calculate 0010 OR 0001 to equal 0011. Type the integer value of the result in the Edit Attribute box; for this example, the value is 3.

1. Click OK.

Enabling change notifications across site links propagates all change notifications. With change notification between sites set, changes propagate to the remote site with the same frequency that they are propagated within the source site, including changes that warrant urgent replication.

**Note:**

Do not enable change notification on demand-dial IP site links or on SMTP site links.

### Forced Replication of a Single Active Directory Object

Slide Objective

Lead In



Support has been added to Windows Server 2003 and Service Pack 4 for Windows 2000 to provide “on demand” replication of a single object. This functionality only applies to domain controllers that are authoritative for the data being replicate (i.e. writable replicas of the naming context). One administrator interface is through the use of an operational attribute known as *replicateSingleObject*. This type of attribute is non-visible and triggers an action based on an LDAP modify command. In this case, the operational attribute is implemented as follows:

From an administrator’s perspective, this kind of replication can be triggered in one of two ways:

1. Using the operational attribute, trigger the replication of a specific object
   1. Determine the GUID of the domain controller that will be used as the source of the replication operation. This can be done by running REPADMIN /showrepl and noting the domain controller object GUID:
   2. Open LDP, bind, and connect to the domain controller that will be the destination of the replication. Select Modify from the Browse menu and use the following parameters:

**Dn**: NULL

**Attribute**: replicateSingleObject

**Values**: <GUID=object\_guid\_of\_source\_machine\_NTDSSettings>**:**DN\_of\_obj\_to\_replicate

*or…*

**Values**: DN\_of\_source\_machine\_NTDSSettings**:**DN\_of\_objt\_to\_replicate

**Operation**: Replace (LDAP\_MOD\_REPLACE)

* 1. Click Enter to move the command into the Entry List box, and then verify that *Synchronous* and *Extended* are both checked. Finally, click *Run* to perform the operation.

1. REPADMIN.EXE implements an interface by providing the /REPLSINGLEOBJ switch.

Replicate Single Object (RSO) is also implemented in the following scenarios:

* When a cross-ref is created during DCPromo for the creation of a new domain, the associated cross-Ref object is created on the Domain Naming Master FSMO role owner, and the machine being promoted enlists RSO to replicate the new cross-reference back to the local computer.
* When a cross-reference is created during the creation of an NDNC, the associated cross-reference object is created on the Domain Naming Master FSMO role owner, and if the machine where the NDNC is created is not the same machine, RSO is implemented to replicate the new cross-Ref back to the local computer.
* When a user’s password is presented to a domain controller and the in the process of validating that password against the PDC for the domain determines that the password on the local machine is incorrect, RSO is used to replicate the change from the PDC to the local computer.