Windows 2000 Server

Chapter 6 - Active Directory Replication

Active Directory™, the directory service that is included with Microsoft® Windows® 2000, is a distributed directory service. Objects in the directory are distributed across the domain controllers in a forest, and all domain controllers in a domain can be updated directly. Replication is the process by which the changes that are made on one domain controller are synchronized with all other domain controllers in the domain or forest that store copies of the same information. Data integrity is maintained by tracking changes on each domain controller and updating other domain controllers in a systematic way. Active Directory replication uses a connection topology that is created automatically, which makes optimal use of beneficial network connections and frees the administrators from having to make such decisions.

In This Chapter

Related Information in the Resource Kit

- For more information about planning sites, site topology, and domain controller location, see "Designing the Active Directory Structure" in the *Microsoft® Windows® 2000 Server Resource Kit Deployment Planning Guide*.
- For more information about the Active Directory database and object storage, see "Active Directory Data Storage" in this book.
- For more information about the directory tree, see "Active Directory Logical Structure" in this book.
- For information about troubleshooting Active Directory replication, see "Active Directory Diagnostics, Troubleshooting, and Recovery" in this book.
- For information about Active Directory replication and restoring domain controllers, see "Active Directory Backup and Restore" in this book.

Active Directory Replication Model

In Active Directory, the directory tree represents all of the objects in a Windows 2000 forest, and this directory tree is partitioned in a way that allows it to be distributed to domain controllers in different domains within a forest. The Active Directory replication model encompasses the manner in which changes are propagated and tracked among domain controllers. Each domain controller in a forest stores a copy of specific parts of the directory. Each defined segment of the directory is called a *directory partition*. A copy of the contents of one directory partition on a specific domain controller is called a *replica*. Updates to replicas are synchronized among the domain controllers that store the same directory partitions during the process of replication.

Directory Partition Replicas

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 (and nowhere else).

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.

Therefore, on a specific 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 controller to another.

Important Note the difference between the directory tree (the Windows 2000 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.

For more information about directory partitions, see "Active Directory Data Storage" in this book. For information about directory searches, see "Name Resolution in Active Directory" in this book. For more information about Global Catalog servers, see "Active Directory Logical Structure" in this book.

Replication Model Benefits

The key benefits of the Active Directory replication model are the following:

- Active Directory always replicates changes to the correct object and is able to differentiate between a deleted object and a new object that has the same distinguished name (also known as "DN"). This is possible because the process of replication is based on the globally unique identifiers (GUIDs) of directory objects, not on their distinguished names.
- This replication model minimizes update conflicts and supports attribute-level conflict detection and resolution. Only those attribute values that have changed, not the entire object, are transmitted as updates.
- Wide area network (WAN) communication is minimized through support for store-and-forward replication and the compression of replication data between sites. Servers contain only a subset of the objects in the entire directory - those that are required for the forest and those that are specific to the server domain. Even though all Global Catalog servers contain all objects in the forest, they contain only a subset of the attributes for these objects.
- Replication topology (including choice of transports) is flexible to make the best use of different network topologies. Sites provide an intelligent basis for automatically generating replication topology and helping clients perform intelligent domain controller selection.
- System configuration remains flexible because the sites are not tied to the partition structure of the directory.

- Speed over high-latency communication links is enhanced because the number of network round-trips by replication protocols is minimized.
- Dependencies on other services, such as time synchronization (W32Time), are minimized.

Replication Model Components

The following mechanisms contribute to the overall replication system:

- Multimaster loose consistency with convergence, which maintains data integrity.
  - "Multimaster" 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" 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.
  - "Convergence" 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

Active Directory uses multimaster replication to accomplish the synchronization of directory information. True multimaster replication can be contrasted with other directory services that use a *master-slave* approach to updates wherein 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 so long that it causes 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. A typical ratio of queries to updates is 99:1. By creating multiple copies of the directory and keeping the copies consistent, the directory service can handle more queries per second.

Multimaster replication provides the following advantages over single-master replication:

- If one domain controller becomes inoperable, other domain controllers can continue to update the directory. In single-master replication, if the primary domain controller becomes inoperable, directory updates cannot take place. For example, if the failed server holds your password and your password has expired, you cannot reset your password and therefore you cannot log on to the domain.
- Servers that are capable of making changes to the directory, which in Windows 2000 are domain controllers, can be distributed across the network and can be located in multiple physical sites.

Store-and-Forward Replication

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. A server that is accepting a change can send the change to nearby servers. One of these servers can then send the change to a distant server, which in turn forwards the change to nearby servers. 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. You administer replication indirectly by defining a simplified network model within your 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

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 a destination unsolicited, 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

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.

Active Directory replication is driven not by logs stored with the source replica, but by the current "state" (the current values of all objects) of the source replica. This state includes information that is used to resolve conflicts and avoid sending the full replica on each replication cycle. Each originating write operation is assigned a sequence number at its originating domain controller. 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 synchronization, and performs fewer database updates because repeated or conflicting updates to an attribute are collapsed into a single state.

Generally speaking, an Active Directory partition replica maintains all of its objects in a list ordered by last modification time. This is a log of sorts, but a coarse one whose size is at most a tiny fraction of the size of the replica itself. 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.

Replication Behavior

Replication behavior 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.

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

- An object is available for replication as soon as it is written. Writes to single objects are atomic, so "partially written" objects are not possible.
- Objects are not necessarily replicated in the order in which they are updated.
- After an update cycle is initiated, a specific replication cycle sends all available changes from the source replica to the destination replica, including changes that occur while the replication cycle is in progress.
- Replication is store-and-forward and "ripples" through a set of connected replicas.
- Multimaster conflict resolution is guaranteed reliable even if clocks become unsynchronized or move backward.
- The graph of replication connections is not always a spanning tree (which by definition does not contain redundant links) - the graph can, and generally does, contain cycles. Redundant connections reduce replication latency, especially in case of failure. A propagation-dampening mechanism eliminates redundant replication.
- Replication within a site is triggered by a change notification mechanism when an update occurs, moderated by a short, configurable delay (because groups of updates frequently occur together).
- Replication between sites typically occurs at scheduled intervals (change notification between sites is optional).
- The system is resilient in the face of load spikes and temporary failures.
- The replication system is designed to be stable. Every time that a replication destination receives information from a replication source, the destination becomes more up-to-date. Recovery from failures involves a minimum of extra work.
- Store and forward replication makes efficient use of WAN links - each update crosses an expensive link only once and is compressed.
- Replication topology is managed automatically and optimizes existing connections.

Microsoft® Exchange Server version 5.5 uses sites and replication somewhat differently than Windows 2000 does. For Exchange administrators who are familiar with Exchange directory replication, Table 6.1 provides a summary of the significant differences between Exchange and Active Directory replication. Treating Active Directory as if it were an Exchange directory does not make optimal use of Active Directory.

Table 6.1 Differences Between Windows 2000 and Exchange Server Version 5.5 Directory Services

Windows 2000 Directory Service	Exchange Server 5.5 Directory Service
Master replicas accept updates independently without communicating with other master replicas.	Each directory service object is mastered in a specific site that can be determined by its distinguished name. Updates are multimaster within the master site.
The basis for replication is the object GUID. When an object is renamed, its GUID does not change, so renaming the object cannot lead to replication errors.	The basis for replication is distinguished names. Therefore, to avoid problems, Exchange does not rename objects.
Replicates an update by transmitting only the changed attributes.	Replicates an update by transmitting the entire object.
Supports compression of replication data between sites over remote procedure call (RPC) or Simple Mail Transfer Protocol (SMTP) transports.	Supports compression of replication data between sites over SMTP transport only.
Supports servers that contain only a subset of the objects in the entire directory in addition to Global Catalog servers, which contain all objects but only a partial set of attributes.	Holds a full replica of the directory in each directory server. (The schema in Exchange is site-specific and is not replicated out of its site.)
Has a flexible replication topology (including choice of transports).	Has a replication topology between sites that is limited to a spanning tree, which cannot contain redundant links. Replication transport between sites is limited to e-mail.
Uses sites to help generate replication topology and to help clients perform intelligent replica selection; but sites are not tied to directory partitioning.	Uses sites to generate replication topology, but sites are also the unit of directory partitioning.

Active Directory Updates

Replication of updates is triggered when a user (or an application or service) updates a particular object or objects on a domain controller. When an update occurs, a timer is started such that changes are collected for a set period, after which the replication engine notifies adjacent domain controllers in the replication topology within the site (in other sites if notification between sites is enabled). After it has been notified that there are changes to be collected, the destination domain controller contacts the source domain controller to request the changes. Replication between sites is typically performed on a scheduled basis: A domain controller requests changes from domain controllers in other sites according to a configurable schedule.

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. A write request either commits and all its effects are durable, or it fails before completion and has no effect. A write request that commits is called an *originating update*. An originating update is initiated and committed at a specific replica. The absolute success or failure of an update applies even for requests, 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 distinguished by the replication system 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 is essentially impossible. Network links fail and clocks drift. And with some systems, unless time is perfectly synchronized among all copies of the directory, there is a chance for data loss or directory corruption.

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.

Some directory services use timestamps as the primary mechanism to determine what updates "win" (are preserved) in a conflict resolution. The typical policy is "last writer wins." Again, this leads to problems when clocks are inaccurate. For example, suppose that you set the clock forward to December 31, 9999, to perform Year 10,000 testing. As part of the testing, you update an attribute of some object. Then you set the clock back again. If the directory service employs a "last writer wins" policy, the update that you performed during the test cannot be overwritten until the year 10,000.

Active Directory replication does not use timestamps as the primary mechanism to determine what 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 set forward to December 31, 9999. With Active Directory, clock skew can never prevent a value from being overwritten.

Deciding What Changes to Replicate: Update Sequence Numbers

The current USN is a 64-bit counter that is maintained by each Active Directory domain controller. 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. This USN value is stored on an updated object in two ways for a replicated write and three ways for an originating write.

- The update's USN value is stored with each attribute changed by the update as the *local USN* of that attribute (originating and replicated writes). You can use the Repadmin command-line tool to view the local USN. Type **repadmin /showmeta *<object_DN>*** at a command prompt and view the column labeled "Loc. USN" in the output. (To use Repadmin, install the Support Tools that are located in the Support\Tools folder on the Windows 2000 Server installation CD. To install the tools, double-click the **Setup** icon in that folder. For information about installing and using the Windows 2000 Support Tools and Support Tools Help, see the file Sreadme.doc in the Support\Tools folder of the Windows 2000 operating system CD. For information about using Repadmin, see *Windows 2000 Support Tools* Help.)
- The maximum local USN among all of an object's attributes is stored as the object's *usnChanged* attribute (originating and replicated writes). You can use Ldp or ADSI Edit to examine an object's *usnChanged* value. Ldp is a graphical tool that you can use to perform connect, bind, search, modify, and delete operations against any LDAP-compliant directory. ADSI Edit is an MMC snap-in that you can install and use to manage Active Directory objects, including viewing and editing attributes. (To use Ldp and ADSI Edit, install the Support Tools that are located in the Support\Tools folder on the Windows 2000 Server installation CD. For more information about using Ldp and ADSI Edit, see *Windows 2000 Support Tools* Help.)
- For an originating write, 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. You can see the originating USN as the column labeled "Org. USN" in the output from the **repadmin /showmeta** command.

Note The *usnChanged* attribute is indexed, which allows objects to be enumerated efficiently in the order of their most recent attribute write.

You can use Ldp or ADSI Edit to read the current USN of a domain controller. These tools use LDAP to read the *highestCommittedUsn* attribute on the rootDSE object for the domain controller. (For information about using ADSI Edit and Ldp to read rootDSE attributes, see "Active Directory Data Storage" in this book.)

High-Watermark

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.

When a destination domain controller requests changes to a directory partition from a source domain controller, the source domain controller provides the changes in increasing order of the *usnChanged* attribute value. The *usnChanged* values from the source domain controller are not stored on objects at the destination domain controller, but the destination domain controller keeps track of the *usnChanged* value of the most recent object it successfully received from the source domain controller for a specific directory partition. This USN is called the destination's high-watermark with respect to the directory partition and the source domain controller.

When requesting changes, 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

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 can contain an entry for each domain controller that holds a full replica of the directory partition. If the up-to-dateness entry that corresponds to source domain controller X contains the USN *n*, the destination domain controller guarantees that it holds all updates to a specific directory partition that originated at domain controller X and that have an originating USN value of less than or equal to *n*.

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.

You can see the up-to-dateness vector in the output of the **repadmin /showvector** command. (For more information about using Repadmin, see *Windows 2000 Support Tools* Help.)

Resolving Conflicts: Stamps

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 a 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* DSA is a GUID that identifies the domain controller that performed the originating write.

You can see all three components of the stamp in output from the **repadmin /showmeta** command. The column labeled "Ver" contains the version, the column labeled "Org. Time/Date" contains the originating time, and the column labeled "Originating DSA" contains the originating DSA (expressed as "site\server" rather than GUID).

When stamps are compared, the version is the most significant, followed by the originating time and then the originating DSA. 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 DSA breaks the tie in an arbitrary fashion.

Two different originating writes of a specific attribute of a particular object cannot assign the same stamp because each originating write advances the version at a specified originating DSA. (Notice that the originating time does not contribute to uniqueness.) Replicated writes cannot decrease the version because values with smaller versions lose during conflict resolution.

For more information about using Repadmin, see *Windows 2000 Support Tools* Help.

Originating Add

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

The Add request fails immediately if the parent object does not exist or if the originating domain controller does not contain a master replica of the parent 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 as follows:

- A Modify request might specify 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.
- A Modify request might specify that a value be added to the current value of an attribute. (This is most useful with an attribute that can have multiple values.) The effect is the same as the equivalent replace request (that is, 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 doesn't change any attributes, the entire request is ignored.

Otherwise, a Modify request computes a stamp 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.

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.

Garbage Collection: Deleting Tombstones

Although they represent deleted objects, tombstones take up space in every directory partition replica; so eventually the tombstones themselves must be deleted to keep the directory database from growing without limit. The garbage collection mechanism deletes tombstones.

Two settings on the object cn=Directory Service,cn=Windows NT,cn=Service,cn=Configuration,dc=ForestRootDomain determine how often garbage collection occurs and which tombstones are deleted:

- The *garbage collection interval* determines the number of hours between garbage collection on a domain controller. The default setting is 12 hours, and the minimum setting is 1 hour.
- The *tombstone lifetime* determines the number of days that tombstones persist before they are vulnerable to garbage collection. The default setting is 60 days, and the minimum setting is 2 days.

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. Garbage collection deletes each tombstone in the set.

The tombstone lifetime must be larger than the worst-case replication latency for any directory partition. Otherwise, a tombstone might be deleted before it has replicated to every directory partition replica. A directory partition replica that fails to replicate the tombstone would never delete the object, so it would be inconsistent with those replicas that had deleted the object.

Tombstone Lifetime Effect on Restore from Backup

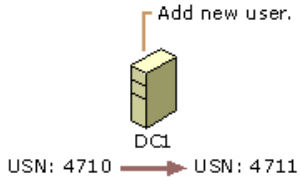
A restore from backup creates a directory partition replica that has not performed replication since the time of backup (or earlier). If the backup was taken more than a tombstone lifetime before the restore, objects deleted in the meantime have no tombstones. The new directory partition replica never learns about these deletions. For this reason, a restore from backup refuses to restore a backup that was taken more than one tombstone lifetime before the time of the restore.

For more information about tombstones and garbage collection, see "Active Directory Data Storage" in this book. For information about restoring Active Directory from backup, see "Active Directory Backup and Restore" in this book.

Tracking Object Creation, Replication, and Change

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

Figure 6.1 shows the replication-related data for the user object when it is first created on domain controller DC1.

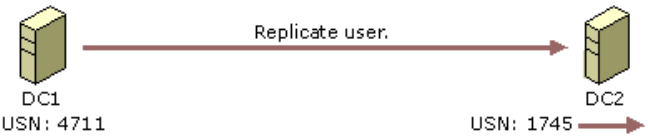


Object: usnCreated: 4711				Object: usnChanged: 4711		
Property	Value	Local USN	Version	Originating Time	Originating DSA	Originating USN
cn	James Smith	4711	1	1999-09-10 10:49.03	<DC1_GUID>	4711
userPassword	6Be8W5q-	4711	1	1999-09-10 10:49.03	<DC1_GUID>	4711
sAMAccountName	JSmith	4711	1	1999-09-10 10:49.03	<DC1_GUID>	4711
userPrincipalName	JSmith@reskit.com	4711	1	1999-09-10 10:49.03	<DC1_GUID>	4711

If your browser does not support inline frames, click here to view on a separate page.

Figure 6.1 Replication-related Data on DC1 When a User Object Is Created

Figure 6.2 shows the change to the destination domain controller when the new user object is replicated. The object is created as a replicated update on DC2. Notice that the per-attribute originating USN and stamp (version, originating time, originating DSA) are replicated from DC1 to DC2, but the per-attribute local USN and per-object *usnChanged* are unique to DC2.

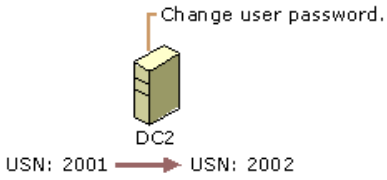


Object: usnCreated: 1746				Object: usnChanged: 1746		
Property	Value	Local USN	Version	Originating Time	Originating DSA	Originating USN
cn	James Smith	1746	1	1999-09-10 10:49.03	<DC1_GUID>	4711
userPassword	6Be8W5q-	1746	1	1999-09-10 10:49.03	<DC1_GUID>	4711
sAMAccountName	JSmith	1746	1	1999-09-10 10:49.03	<DC1_GUID>	4711
userPrincipalName	JSmith@reskit.com	1746	1	1999-09-10 10:49.03	<DC1_GUID>	4711

If your browser does not support inline frames, click here to view on a separate page.

Figure 6.2 Replication-related Data on DC2 When a New User Object Is Replicated from DC1

Figure 6.3 illustrates the change in the replicated object on DC2 when someone changes the password (*userPassword* propertyin the diagram) of the object on that domain controller. By this time, the current USN on DC2 has increased from 1746 to 2001. The update request that changes the password increments the current USN to 2002 on DC2. The request also sets the password attribute's originating USN and local USN to 2002 and creates a new stamp for the password value. The version number of this password's stamp is 2, which is one version number higher than the version of the previous password.

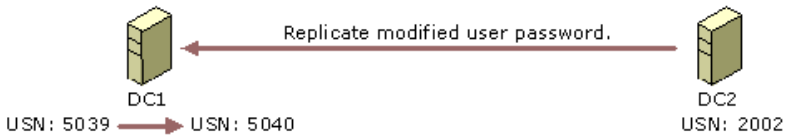


Object: usnCreated: 1746				Object: usnChanged: 2002		
Property	Value	Local USN	Version	Originating Time	Originating DSA	Originating USN
cn	James Smith	1746	1	1999-09-10 10:49.03	<DC1_GUID>	4711
userPassword	Secaucus2	2002	2	1999-09-10 11:53.29	<DC2_GUID>	2002
sAMAccountName	JSmith	1746	1	1999-09-10 10:49.03	<DC1_GUID>	4711
userPrincipalName	JSmith@reskit.com	1746	1	1999-09-10 10:49.03	<DC1_GUID>	4711

If your browser does not support inline frames, click here to view on a separate page.

Figure 6.3 Replication-related Data on DC2 After the User Password Value Has Been Changed on DC2

In Figure 6.4, the changed password is now replicated back to the original domain controller, whose current USN has increased to 5039. The replicated update increments the current USN to 5040. The per-attribute originating USN and stamp (version, originating time, originating DSA) are replicated from DC2 to DC1, and the per-attribute local USN and per-object *usnChanged* values are set to 5040.



Object: usnCreated: 4711				Object: usnChanged: 5040		
Property	Value	Local USN	Version	Originating Time	Originating DSA	Originating USN
cn	James Smith	4711	1	1999-09-10 10:49.03	<DC1_GUID>	4711
userPassword	Secaucus2	5040	2	1999-09-10 11:53.29	<DC2_GUID>	2002
sAMAccountName	JSmith	4711	1	1999-09-10 10:49.03	<DC1_GUID>	4711
userPrincipalName	JSmith@reskit.com	4711	1	1999-09-10 10:49.03	<DC1_GUID>	4711

If your browser does not support inline frames, click here to view on a separate page.

Figure 6.4 Replication-related Data on DC1 After the Password Change Has Been Replicated to DC1

Propagation Dampening

Multiple replication paths can exist between a pair of domain controllers. Multiple paths provide fault tolerance and can reduce latency. However, when multiple paths exist, you might expect the same change to be sent along each path to a specific domain controller. You might even be concerned that a change might replicate in an endless loop. Active Directory prevents these potential problems with multiple paths by using the up-to-dateness vector described earlier.

The following is an example of how ordinary replication occurs:

1. A updates a password attribute. The originating USN of the attribute is set to 3.
2. Destination DC B requests changes from source DC A and sends its high-watermark and up-to-dateness vector to DC A.
3. According to the high-watermark that was passed by DC B, source DC A examines one or more objects, one of which contains the changed password. When DC A encounters the changed password attribute, it proceeds as follows:

- First, DC A finds that the originating DSA of the password attribute is DC A.
- Therefore, DC A reads the up-to-dateness vector supplied by DC B and finds that DC B is guaranteed to be up-to-date with updates that originated at DC A and that have an originating USN of less than or equal to 2.
- DC A then finds that the originating USN of the password attribute is 3.
- Because 3 is greater than 2, DC A sends the changed password attribute to DC B.

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

would not send the changed password to DC B.

Multimaster Conflict Resolution Policy

As described in "Resolving Conflicts: Stamps" earlier in this chapter, Active Directory resolves attribute value conflicts by comparing versions. However, attribute value conflicts are not the only conflicts that arise.

Given the semantics of LDAP directories, there are four possible conflicts that are created by multimaster updates. Two of these conflicts are in fact different sides of the same conflict, reducing the number of conflict situations to the following three:

- *Attribute value*. 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.
- *Add or Move under deleted parent, Delete* non-leaf object. An Add or Move operation makes object C a child of object P. Concurrently, at another domain controller, a Delete operation deletes object P.
- *Sibling name conflict*. An Add or Move operation makes C1 a child of P with C1.rdn = R. Concurrently, at another domain controller, an Add or Move operation makes C2 a child of P with C2.rdn = R.

These conflicts can occur in any multimaster LDAP directory.

In Active Directory, the general approach to resolving these conflicts is to order all update operations (Add, Modify, Move, and Delete) by assigning a globally unique (per-object and per-attribute) stamp to the originating update. In the case of a conflict, the ordering of stamps allows a consistent resolution. This approach is applied to the three preceding conflicts as follows:

- Attribute value. Let V be the value among {V1, V2} with the larger stamp. After resolution, the attribute value at all domain controllers is V.

Note Conflicts are resolved at the level of the entire attribute value, even for multivalue attributes. The value after resolution is one multivalue or the other, not some combination.

- Add or Move under deleted parent, Delete non-leaf object. After resolution, at all replicas, object P is deleted and object C is a child of the special LostAndFound container in the directory partition. Stamps are not involved in the resolution.
- Sibling name conflict. Let C be the object among {C1, C2} whose relative distinguished name attribute value has the smaller stamp. After resolution, at all domain controllers, C.rdn is a system-assigned value unique to C that cannot conflict with any client-assigned value. For example, if the relative distinguished name of object C was "ABC" before conflict resolution, its relative distinguished name after resolution is "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.

Replication Topology

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 favorable 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.

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

Topology-related Components

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 sites and the domain controllers that are associated with them.
- The connections that identify the routes that replication takes between domain controllers within sites.
- The links that make replication connections between sites possible.
- The transports that the links use to communicate between sites.

The KCC uses these and other objects and their properties to create and manage the connections by which directory updates are transferred 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 (every 15 minutes, by default), these partnerships are added, removed, or modified automatically, as necessary, on the basis of what domain controllers are available and how close they are to each other on the network.

The KCC uses the following components to manage replication:

Connections The KCC creates connections that enable domain controllers to replicate with each other. A connection defines a one-way, inbound route from one domain controller, the source, to another domain controller, the destination. The KCC reuses existing connections where it can, deletes unused connections, and creates new connections if none exist that meet the current need.

Servers Each domain controller is represented by a server object. The server has a child object, NTDS Settings, which stores the

inbound connections; that is, the connection objects for a server designate the connections *from* source domain controllers *to* the server object.

Sites Sites define sets of domain controllers that are well connected in terms of speed and cost. Domain controllers in the same site replicate on the basis of notification: when a domain controller has changes, it notifies its replication partners. Then the notified partner requests the changes, and replication takes place. Because there is no concern about replication speed or cost, replication within sites occurs as needed rather than as scheduled.

Note To allow for the possibility of network failure, which might cause one or more notifications to be missed, a default schedule of once per hour is applied to replication within a site, in addition to change notification.

Replication between sites occurs according to a schedule; you can use the schedule to determine the most beneficial time for replication to occur on the basis of network traffic and cost. A site is the equivalent of a set of one or more Internet Protocol (IP) subnets.

Note Under circumstances where connections cannot be initiated between both sites (for example, when one site requires a dial-up connection), reciprocal replication can be initiated on the basis of changes rather than a schedule. (For more information about reciprocal replication, see "Reciprocal Replication" later in this chapter.)

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. For more information about subnets, see "Introduction to TCP/IP" in the *Microsoft® Windows® 2000 Server Resource Kit TCP/IP Core Networking Guide*.

Site Links For replication to occur between two sites, a link must be established between the sites. Site links are not generated automatically and 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. 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 equally well connected.

Bridgehead Servers To communicate across site links, the KCC automatically designates a single server, called the *bridgehead server*, in each site to perform site-to-site replication. Subsequent replication occurs by replication within a site. When you establish site links, you can designate the bridgehead servers that you want to receive replication between sites. By designating a specific server to receive replication between sites, rather than using any available server, you can specify the most beneficial conditions for the connection between sites. Bridgehead servers ensure that most replication occurs within sites rather than between sites.

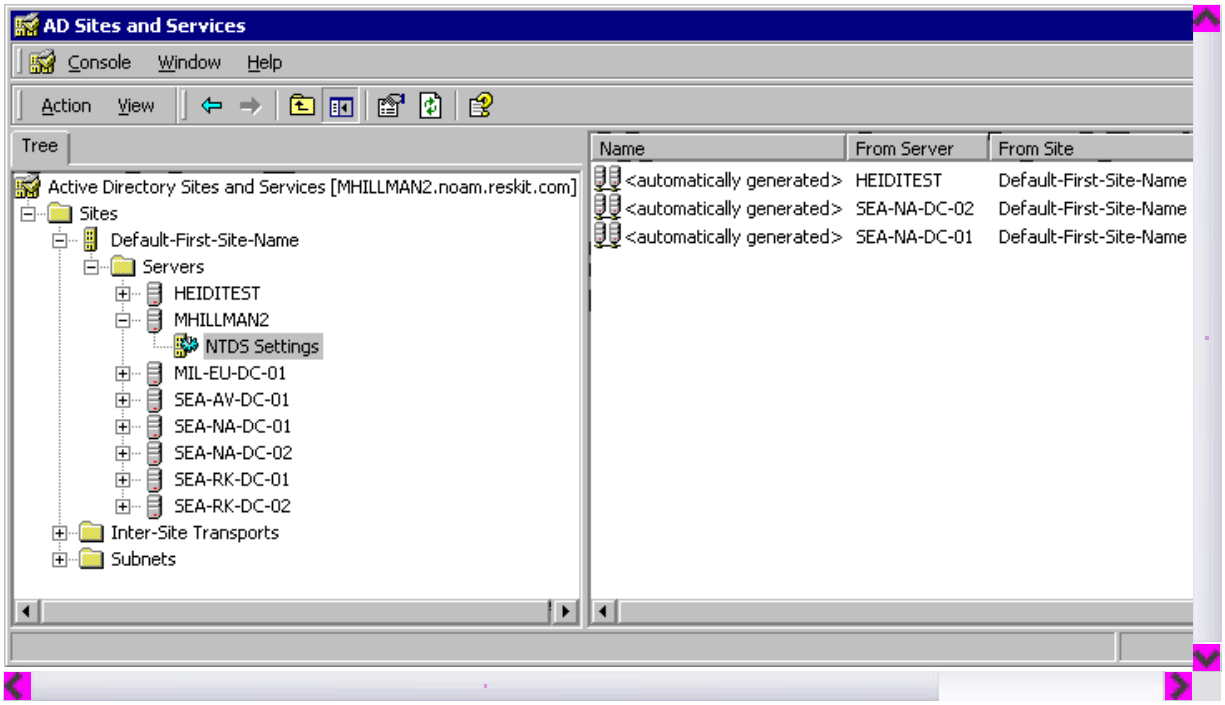
Site Link Bridges When more than two sites are linked for replication and use the same transport, all of the site links are "bridged" in terms of cost by default, assuming that the site links have common sites. When site links are bridged, they are *transitive*. That is, all site links for a specific 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.

Sites Container Hierarchy in Active Directory

Active Directory Sites and Services is the administrative tool that you use to view and manage the hierarchy of objects that are used by the KCC to effect the replication topology. The hierarchy is displayed as the contents of the Sites container, which is a child of the Configuration container. The distinguished name of the Sites container is cn=Sites,cn=Configuration,dc=<*ForestRootDomain*>. The Configuration container is the topmost object in the configuration directory partition and the Sites container is the topmost object in the hierarchy of objects that are used to manage and implement Active Directory replication. The Sites container hierarchy contains the following objects:

- The Sites container, which contains an object for each site in the forest. In addition to site objects, Sites also contains the Subnets container, which contains subnet definitions in the form of subnet objects. Each subnet object has a *siteObject* attribute that links it to a site object.
- Site objects, each of which contains two child objects:
  - The NTDS Site Settings object, which stores directory properties common to all domain controllers in the site, including the schedule for replication within the site.
  - The Servers container, which stores a server object for each domain controller in the site.
- Server objects, each of which contains an NTDS Settings object.
- NTDS Settings object, which represents an instantiation of Active Directory on that server. (When Active Directory is removed from a server, its NTDS Settings object is deleted from Active Directory, but its server object remains.) 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.
- Connection objects, each of which represents a unidirectional replication agreement between two specific domain controllers, where the destination domain controller is the server object that is the parent object of the NTDS Settings object that stores the connection. Connection objects are created automatically by the KCC; they can also be created manually.

Figure 6.5 shows the expanded Sites container. The NTDS Settings object for one server is selected, and the related connection objects are displayed in the details pane. The **From Server** column displays the name of the domain controllers from which the selected domain controller receives replication (the names of its current replication partners).



If your browser does not support inline frames, click here to view on a separate page.

Figure 6.5 Hierarchy of Objects in the Sites Container

For information about how to create and manage objects in the Sites container, see Windows 2000 Server Help.

Sites and Replication

A *site* represents 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 nor 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 Efficiency

Replication within a site is driven by changes. The high speed and reliability of LAN connectivity lends itself to on-demand data transfer. When a change is applied to a specific replica, the replication engine is triggered. The replication engine waits for a configurable interval (by default, five minutes) and then notifies the first replication partner. Each additional partner is notified after a configurable delay (by default, 30 seconds).

Generation of the topology within a site is achieved by using the minimum number of connections possible. The default replication topology in a site is a bidirectional ring, with sufficient additional connections added to keep the number of hops between replication partners to three or less. To summarize, the default worst-case delay for propagation within a site is the maximum number of hops between a source and destination domain controller (3 hops) multiplied by the delay at each domain controller in the path (5 minutes), or 15 minutes for three hops.

Replication between sites is scheduled so that you can control replication costs and keep replication from overwhelming your communication links. Several factors contribute to the speed of replication between sites, including the nature of the physical connections and how expensive they are to use. Active Directory supports site connections in all speed ranges, from T3 links on the high end to dial-up lines on the low end. You use the speed of the connection between your sites to assign a cost to the communication link, and replication uses the cost to establish the least expensive route for replication traffic.

For information about planning networks, see "Determining Network Connectivity Strategies" in the *Deployment Planning Guide*.

Site Design with Replication in Mind

When you group 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 conserve network bandwidth.
- 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 the arrival of updates, which thereby 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 name of a domain

directory object does not reflect the site or sites in which the object is stored.

Note In the Exchange Server directory service, sites are tied to the namespace.

The hierarchy in the Sites container reflects object names within the configuration directory partition, where site names do appear in the distinguished names of objects in the Configuration container hierarchy, as seen in Active Directory Sites and Services.

For information about planning sites and site topology for Active Directory, see "Designing the Active Directory Structure" in the *Deployment Planning Guide*.

Subnet-to-Site Mapping

When you plan your sites, you decide what subnets provide the best connectivity for replication and add them to the same site. You can use Active Directory Sites and Services to create subnets, and then you can create a site and associate the subnets with the site. You create a site by creating a site object in Active Directory and then defining a set of one or more subnets that belong to the site.

The determination of whether a computer is in a site is that its IP address maps to a subnet object in Active Directory. In the default directory, 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, you can specify the network addresses that are provided 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 information about subnets and subnet masks, see "Introduction to TCP/IP" in the *TCP/IP Core Networking Guide*. For information about creating subnets and assigning subnet masks, see Windows 2000 Server Help. For information about designing your network, see "Determining Network Connectivity Strategies" in the *Deployment Planning Guide*. For information about designing sites for Active Directory, see "Designing the Active Directory Structure" in the *Deployment Planning Guide*.

When to Define a New Site

When you have slow links between network segments, it is recommended that you create two sites and place domain controllers into the sites according to the following general rules:

- Deploy at least one Global Catalog per site.
- Deploy DNS servers on a site level.

The first domain controller in the forest is designated automatically as a Global Catalog server. When you create additional sites, you can use Active Directory Sites and Services to select the Global Catalog option in the properties for the NTDS Settings object of the server that you want to be the Global Catalog. Having a Global Catalog server in each site improves search performance because searches do not have to cross site boundaries. In addition, a Global Catalog server is required for logging on to the domain; if a connection between sites is not available, logging on is not possible.

Note If a Global Catalog server is not available in one site but there is another Global Catalog server in a remote site, the server in the remote site can be used for the logon process. If no Global Catalog is available in any site, the logon process proceeds with cached logon information.

The availability of DNS directly affects the availability of Active Directory. Clients rely on DNS to be able to find a domain controller, and domain controllers rely on DNS to find other domain controllers. As a general rule, you configure at least one DNS server in every site.

When you create sites and put servers into the sites, you connect the sites with site links and configure the links to reflect the network characteristics in terms of how often you want replication to occur on the link.

For information about planning sites and about domain controller, DNS server, and Global Catalog server placement, see "Designing the Active Directory Structure" in the *Deployment Planning Guide*. For information about Global Catalog servers, see "Active Directory Logical Structure" in this book.

Default Site

When you install Active Directory on the first domain controller in the site, an object named Default-First-Site-Name is created in the Sites container. The first domain controller is necessarily installed into this site. Subsequent domain controllers are either installed into the site of the source domain controller (assuming the IP address maps to the site) or installed directly into an existing site. When your first domain controller has been installed, you can rename Default-First-Site-Name to the name that you want to use for the site.

When you install Active Directory on subsequent servers, if alternative 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. Otherwise, it is added to the site of the source domain controller.

For information about how sites are identified for new domain controllers, see "Active Directory Data Storage" in this book.

Server and Site Connections

Replication occurs between all domain controllers in the same site and between bridgehead servers in different sites. The replication connections between domain controllers in the same site are always created automatically by the KCC (and also can be created manually, although there is usually no need to do so). When you have more than one site, you create site links to connect them. When two sites are connected by a site link, the KCC also creates connections between the two bridgehead servers - one in each site for each domain that has domain controllers in the site. If there are multiple domains per site and each domain has domain controllers in more than one site, there are multiple bridgehead servers per site because domain controllers in the same domain must have connections that include only servers in that domain. The KCC selects the bridgehead servers automatically, or you can designate them manually if you want the same servers to be always used as bridgehead servers.

Replication transport protocols determine the manner in which replication data is transferred over the network media. Your network environment dictates the transports that you can use.

Server Objects

When you install Active Directory on a Windows 2000 Server-based computer, the installation process creates a server object that represents the domain controller. A server object is distinct from the computer object that represents the computer as a security principal. These objects are in separate directory partitions; 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.

Server objects are children of site objects. The parent site of a server must contain the server subnet. If a domain controller's IP address or the subnet-to-site associations are changed after Active Directory is installed on the server computer, the domain controller does not change sites automatically. It must be moved to the new site manually if that site is the desired location.

Make sure that you create the site link before you move a domain controller to a different site. When replication between sites uses the

SMTP transport, it is especially important that the domain controller first be properly configured within the site with RPC over IP connectivity.

Note When you make configuration changes that affect replication, the configuration changes replicate by using the old replication settings until they reach the systems where they take effect. Replication between sites does not occur until the following events occur:

- The site link is created somewhere in the directory.
- This change has replicated to the system in the site that is responsible for creating the intersite topology. (For more information about intersite topology generation, see "Automated Intersite Topology Generation" later in this chapter.)
- The KCC on that system has run and created the new connections.
- The new connections have replicated to the bridgehead server.
- The KCC on the bridgehead server has run and has translated the connections into replication partner relationships.
- Replication occurs on the bridgehead server.

Server Connections

The KCC creates connections for every domain controller that has a server object in the Sites container. These objects specify a one-way communication to the current system from another system. 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 - that is, it represents the inbound half of a connection. The connection object contains a replication schedule and specifies a replication transport.

Connection objects are created in two ways:

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

A connection is unidirectional; a bidirectional replication connection is represented as two connection objects under two different NTDS Settings objects.

Ownership of Connection Objects

Connections that are created automatically by the KCC are "owned" by the KCC. Likewise, if an administrator creates a new connection or modifies an existing connection, that connection is owned by the administrator. If a connection object is not owned by the KCC, the KCC does not modify it or delete it. The implication of creating or modifying a connection object is that you want the object to remain as you define it. Ownership protects the object from being changed by the KCC.

Note Ownership of a connection object does not affect security access to the object; it determines only whether the KCC can modify or delete the object.

Ownership of a connection object is determined by the value in the *options* attribute on the connection object. If this value (*value* BITWISE-AND 1) equals 1, the KCC owns the connection. If you modify a KCC-generated connection, the *options* value changes. If you create a new connection object, the value of the *options* attribute is set to 0.

You can take ownership of a KCC-generated connection object by using Active Directory Sites and Services to modify the connection. Click the **Change** button to modify object properties; when the changes replicate, the *options* attribute value changes to reflect that the KCC does not own the object. For more information about creating a connection object, see Windows 2000 Server Help.

Note It is usually not necessary to create manual connections when the KCC is being used to generate automatic connections. The KCC automatically reconfigures connections as conditions change. Adding manual connections when the KCC is employed potentially increases replication traffic by adding redundant connections to the optimal set chosen by the KCC. When manually generated connections exist, the KCC uses them wherever possible.

Connection Schedule

Each connection object has a schedule that is set automatically by the KCC when it determines the best route for replication. The connection schedule controls how frequently periodic replication occurs. The connection schedule has a minimum increment of 15 minutes. The default setting for replication within a site is once per hour, which you can change by modifying the NTDS Site Settings object. By using Active Directory Sites and Services, you can set a schedule of None (no replication), once per hour (default), twice per hour, or four times per hour.

Note The Active Directory Sites and Services user interface confines the settings to once every 15 minutes for a specific hour or hours (from 1 hour to 24 hours) during the week. By using ADSI Edit or scripts, you can set replication on or off independently during each 15-minute window in the week.

The connection schedule receives its default from the schedule on the NTDS Site Settings object. When the KCC creates a new connection, the connection receives the schedule that is set on the NTDS Site Settings object. To override the default on the NTDS Site Settings object, you can manually set a schedule on the connection objects.

Note This manual override is effective only for manually created objects. If you attempt to update the schedule on a KCC-owned object, the KCC reverts to the schedule on the NTDS Site Settings object the next time it runs.

For a connection between sites, the KCC derives the schedule from the site link schedule, which controls how frequently the site link is active. The default setting for replication between sites is three hours, which you can change by modifying the associated site link object or objects.

Within a site, replication is triggered by changes; if a change occurs, replication is initiated (the domain controller with changes sends a notification to its replication partners that it has changes) after a default interval of five minutes. When no changes occur on the domain controller during the time allowed by the connection object schedule, replication is triggered (the domain controller requests changes from its replication partners) on the basis of the schedule (by default, once per hour).

The schedule is most important in replication between sites, where it is the primary mechanism that triggers replication. Because the granularity of the connection object schedule is 15 minutes, you cannot schedule replication between sites to occur any more frequently than once every 15 minutes.

Connecting Directory Partitions

Replication is performed between directory partition replicas, and connections are created between the servers to accommodate replication of as many partitions as the two servers have in common. Two domain controllers in the same forest always have at least two directory partitions in common: the configuration directory partition and schema directory partition. If the domain controllers are in the same domain, they also have a domain directory partition in common. If the domain controllers are Global Catalog servers for the same forest, they have partial domain directory partitions for every domain in common; if they are in the same domain, they have a full domain directory partition in common as well.

If a connection exists from one domain controller to another, it is used for replicating as many directory partitions as are common to the two servers. There is never a need for the KCC to create multiple connections linking the same two domain controllers in the same direction.

Site Links

When you have more than one site in your replication system, you must create links to connect the sites for replication. In Active Directory, a site link object identifies a set of sites that can be scheduled to communicate at uniform cost through some transport between sites. Site links communicate the connectivity that the KCC assumes between sites. Usually, you name site links for the sites that they connect. For example, if you have a site named "New York" and a site named "London," you might name your site link "NY-London."

Site links also specify the cost of the link, which controls the desirability of remote sites as sources of replication information. In addition, site links specify the schedule, how frequently periodic replication occurs over this link during the schedule window, and the behavioral options that might influence how replication occurs.

By default, site links for the same IP transport that have sites in common are bridged by site link bridges, which enable the KCC to treat the set of associated site links as a single route.

Bridgehead Servers

A "bridgehead" is a point where a connection leaves or enters a site. Servers that have connection objects for connections between sites are called *bridgehead servers.* 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. The KCC generates the connections and thereby causes the domain controllers that store the connections between sites to act as bridgeheads in the topology.

If you want to limit the KCC's choices of servers that it can designate as bridgeheads (that is, restrict the domain controllers in which the KCC can create connections between sites), select one or more domain controllers in the site that you want the KCC to always consider as "preferred" bridgehead servers. These servers are used exclusively to replicate changes collected from the site. If you specify preferred bridgehead servers, be aware of the following consequences:

- You limit the ability of the KCC to fail over to another bridgehead server when the designated server is down.
- You must assign one bridgehead server for each domain and writable directory partition combination in your forest, the cost of which might be prohibitive in a large organization.

Replication Transports

Replication transports provide the wire protocols that are required for data transfer. Windows 2000 provides three levels of connectivity for replication of Active Directory information:

- Uniform high-speed, synchronous 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 you create a site link object, you 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).

Synchronous vs. Asynchronous Communication

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. Therefore, at any particular time, a domain controller has a maximum of one synchronous request outstanding.

In the case of asynchronous replication, the destination domain controller does not wait for the reply, and it can have multiple asynchronous requests outstanding at any particular time. In synchronous transmission, therefore, the reply is received in a short amount of time; in asynchronous transmission, the reply is not necessarily received in a short time.

Transport for Replication Within a Site

All replication within a site occurs over synchronous RPC over IP transport. The focus for replication within a site is the fast, uncompressed delivery of updates. Replication events occur more frequently within a site than between sites, and the overhead of compression would be inefficient over fast connections.

Note Replication between sites is compressed.

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 locator 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 therefore never needs to know which port to use for Active Directory replication.

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

Active Directory registers the endpoint when it starts, and it uses either a dynamically assigned port or a specified port, whichever you have configured. To specify a fixed port for routed environments or where port filtering is employed, you can add or modify the **TCP/IP Port** entry in HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \NTDS \Parameters to configure Active Directory to register a specific port with the endpoint mapper. This value can be any valid TCP/IP port number.

To set the TCP/IP Port entry

1. In a registry editor, navigate to HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \NTDS \Parameters.
2. Double-click the **TCP/IP Port** entry, and assign a valid port number.
3. Close the registry editor.

Caution Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. There are programs available in Control Panel or Microsoft Management Console (MMC) for performing most administrative tasks. These programs provide safeguards that prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Registry editors bypass the standard safeguards that are provided by these administrative tools. Modifying the registry is recommended only when no administrative tool is available. Before you make changes to

the registry, it is recommended that you back up any valuable data on the computer. For instructions about how to edit registry entries, see Help for the registry editor that you are using. For more information about the registry, see the *Microsoft Windows 2000 Resource Kit* Technical Reference to the Windows 2000 Registry (Regentry.chm).

Transports for Replication Between Sites

Windows 2000 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 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.

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:

- 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 hourly 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.

The Intersite Messaging (ISM) service allows for multiple transports to be used as add-ins to the ISM architecture. ISM provides services to the KCC in the form of querying the available replication paths. ISM enables messaging communication that can use SMTP servers other than those that are dedicated to processing e-mail applications such as Exchange Server.

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 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 frames, 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 for either, complete retransmission is necessary.
- If bandwidth is limited, the same TCP retransmission characteristics apply. (RPC time-out is much longer than TCP time-out.)

Because SMTP is not used for replication of domain directory partitions, Windows 2000 provides point-to-point synchronous RPC replication in addition to asynchronous SMTP replication between sites to allow the flexibility of having domains span multiple sites. RPC is best used between well-connected sites 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 companies that have a network backbone that is not based on TCP/IP, such as companies 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 to deliver streams of mail in each direction. Thus, RPC transport expects a response to any request more or less 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.

Benefits of SMTP Replication Between Sites

Although SMTP replication usually is slower than RPC, there are conditions that call for SMTP replication exclusively, and others that favor it, such as the following:

- RPC-based replication is currently synchronous, whereas SMTP is asynchronous. Where bandwidth is limited, it can be disadvantageous to force an entire transaction to complete before another can begin. With SMTP, several transactions can be processing simultaneously so that each transaction is receiving some attention most of the time, as opposed to no attention for prolonged periods, which can result in RPC time-outs.
- SMTP traffic can be secured, monitored, and managed across a WAN environment.
- Where end-to-end online IP connectivity is impossible - for example, A can communicate with B, and B can communicate with C, but A can never communicate with C directly - mail can be used and routed A to B, B to C, C to B, or B to A.

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: >=1
  - Replicator intra site packet size (bytes)
  - Range: >=10 KB
- For RPC replication between sites:
  - Replicator inter site packet size (objects)

- Range: >=1
- Replicator inter site packet size (bytes)
- Range: >=10 KB

- For SMTP replication between sites:
  - Replicator async inter site packet size (objects)
  - Range: >=1
  - 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.

Caution Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. There are programs available in Control Panel or Microsoft Management Console (MMC) for performing most administrative tasks. These programs provide safeguards that prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Registry editors bypass the standard safeguards that are provided by these administrative tools. Modifying the registry is recommended only when no administrative tool is available. Before you make changes to the registry, it is recommended that you back up any valuable data on the computer. For instructions about how to edit registry entries, see Help for the registry editor that you are using. For more information about the registry, see the *Microsoft Windows 2000 Resource Kit* Technical Reference to the Windows 2000 Registry (Regentry.chm).

Managing Replication Between Sites

Although replication within sites is optimized for LAN connectivity and requires little or no management, you have control over when and how replication between sites occurs. You want to maximize efficiency and minimize cost, and there are decisions that must be made on the basis of your network environment, physical location, and business needs. The KCC generates the intersite topology automatically, but the settings on the site links are the factors that the KCC considers in the process.

When multiple sites participate in the replication topology of domain controllers in the same forest, the default intersite topology is a least-cost spanning tree, where "cost" is administratively set to favor various routes. Replication between sites can occur synchronously by RPC over IP transport or asynchronously by SMTP over IP transport.

Note A spanning tree algorithm is applied to network connections to eliminate redundant routes and thereby reduce consumption of expensive WAN network bandwidth.

Planning Replication Between Sites

Replication within sites requires little or no planning because it is fully automatic. However, when you have multiple sites, the following steps can be used to plan how replication occurs between them:

1. Identify sites that are well connected through backbones, and create low-cost site links between these sites.
2. Identify sites that are all connected to each other with a comparable transport, and create medium-cost site links between them - for example, full mesh links (remote sites that are connected over telecommunication links), frame relay cloud links (a point-to-point system that uses a private virtual circuit), medium area network (MAN) links with T1 connections.
3. Identify remaining WAN links.
4. Create a site link for each pair of sites that cross a WAN link.
5. Create a schedule that meets user needs.

Avoid high-frequency times. Site links must have windows of time in common that are available for routing.

For information about planning sites and site topology for Active Directory, see "Designing the Active Directory Structure" in the *Deployment Planning Guide*.

Managing Site Links

Connection objects are created automatically by the KCC for replication both within a site and between sites. For connection objects to be created between two sites, however, you must manually create a link that connects the two sites. These links, implemented through site link objects in Active Directory, identify the transport protocol and scheduling required to replicate between two sites. Administrators use Active Directory Sites and Services to create the site links, and the KCC creates the connections accordingly when it generates the intersite topology.

Site link objects can be created in two transport-specific containers within the Inter-Site Transports container in Active Directory Sites and Services. By creating the link in one or the other container, you associate the link with the respective replication transport. The Inter-Site Transports container is a child of the Sites container, and it also has child containers:

- The IP container, which contains site link objects that use RPC over IP synchronous replication transport.
- The SMTP container, which contains site link objects that use SMTP over IP asynchronous replication transport.

When the KCC configures the connection objects for replication between sites, it takes the settings on the site link object into account to create the best connection. For example, one of the site link settings is the cost of the connection. When it has a choice, the KCC chooses a remote site whose link has the lowest cost when it forms connections.

For IP transport, a typical site link connects only two sites and corresponds to an actual WAN link. An IP site link connecting more than two sites might correspond to an ATM backbone that connects, for example, more than two clusters of buildings on a large campus or connects several offices in a large metropolitan area that are connected by leased lines and IP routers.

A site can be connected to other sites by any number of site link objects. Each site in a multi-site directory must be connected by at least one site link. Otherwise, it cannot replicate with domain controllers in any other site, so the directory is disconnected. Therefore, if there is more than one site in the forest, you must configure at least one site link.

Figure 6.6 shows two sites that are connected by one site link. A single domain has domain controllers in both sites. When topology generation occurs, connection objects between bridgehead servers in the site are created by the KCC and replication occurs according to the settings on the site link.

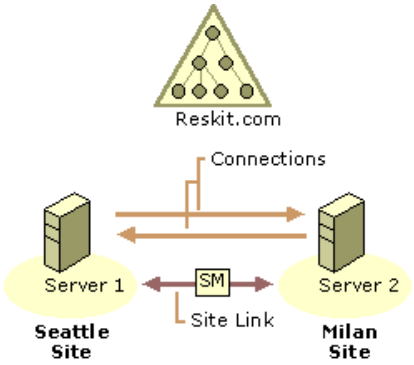


Figure 6.6 Two Sites That Are Connected by a Site Link

Figure 6.7 shows three sites connected by two site links. By default, site links are transitive. Therefore, replication messages can flow from the Atlanta site, through the Seattle site, and on to the Milan site. In this scenario, because the Seattle site contains a full replica of reskit.com, there is no need for direct replication between Milan and Atlanta; all replication between them is transitive through the Seattle site.

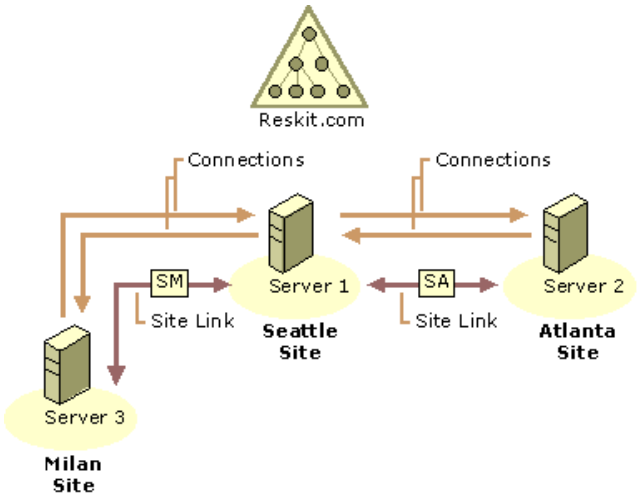


Figure 6.7 Three Sites That Are Connected by Two Site Links

Site Link Settings

In Active Directory Sites and Services, the **General** tab in the **Site Link Properties** dialog box contains the following options for configuring site links to control the replication topology:

- A list of two or more sites to be connected.
- A schedule that determines during what time periods the link is available for replication. For example, you might schedule a site link for a dial-up line to be available during off hours (when telephone rates are low) and unavailable during high-cost regular business hours.

Note Scheduling information is ignored by site links that use SMTP transports; the mail is stockpiled and then exchanged at the times that are configured for your mail infrastructure.

- Cost, a single numeric cost factor associated with communication over the link. Higher cost numbers represent more expensive messages. For example, sites that are connected by low-speed or dial-up connections would have high-cost site links between them. Sites that are well connected through backbone lines would have low-cost site links. Where multiple routes or transports exist between two sites, the least expensive route and transport are used.
- Period, an interval in minutes that determines how often replication can occur (default is 180 minutes, or 3 hours). The minimum period is 15 minutes.

The site link settings let you control replication topology and timing independently:

- You control topology by setting the costs on site links. In a common scenario, you might set cost = 1 for site links that are part of your backbone network, and cost = 100 for site links that correspond to slow connections to branch offices. Setting costs in this way ensures that a branch office replicates with a domain controller in a site that is part of the backbone, never directly with a second branch office. These cost numbers have no influence on the replication period.
- You control the replication period by setting an interval in minutes on site links.
- You control link availability by setting a schedule on site links. You might use the default (100 percent available) schedule on most links, but block replication traffic during peak business hours on links to certain branches. By blocking replication, you give priority to other traffic, but you also increase replication latency.

Cost Factor

You assign cost values to site links to favor inexpensive connections over expensive connections. The costs of providing bandwidth is a factor that is to be taken into account when you define site boundaries; it is recommended that these costs be defined on a sitewide basis. Cost is usually based not only on the total bandwidth of the link but also on the availability, latency, and monetary cost of the link.

For example, a 128-kilobits per second (Kbps) permanent link might be assigned a lower cost than a dial-up 128-Kbps dual ISDN link because the dial-up ISDN link has replication latency-producing delay that occurs as the links are being established or removed. Furthermore, in this example, the permanent link might have a fixed monthly cost, whereas the ISDN line is charged according to actual usage. Because the company is paying up-front for the permanent link, the administrator might assign it a lower cost than the ISDN line. The ISDN connections in this example only add extra monetary cost to the already paid-for permanent line.

Table 6.2 shows the speeds for different types of networks; you can use these network speeds to estimate cost.

Table 6.2 Network Speeds for Estimating Cost

Network Type	Speed
Very slow	56 Kbps
Slow (typical in Europe)	64 Kbps
ISDN	64 Kbps or 128 Kbps
Frame relay	Variable rate, commonly between 56 Kbps and 1.5 megabits per second (Mbps)
T1	1.5 Mbps
T3	45 Mbps
Asynchronous Transfer Mode (ATM)	Variable rate, commonly between 155 Mbps and 622 Mbps
Gigabit Ethernet	1 gigabit per second (Gbps)

Before you assign any costs, define a model for your WAN. On the basis of the cost plus other factors (availability and replication latency), you can establish a set of costs that can be implemented throughout the forest. Where a cost is assigned, it must always mean the same thing in any other place where the same cost is assigned. Table 6.3. shows an example of the cost breakdown in a forest for a network where a high speed has a lower cost.

Table 6.3 Example of Cost Breakdown for a Forest on a High-Speed Network

Network Type	Cost Value
T1 to backbone	1
56-kilobit link	500
Branch office	1,000
International link	5,000

The **Cost** setting on a site link provides a relative value for the cost of communication between all sites that are part of the link. (By default, site link settings are *transitive* between the sites that they connect.) For example, if you create an IP site link object XYZ that connects the sites X, Y, and Z with cost 5, you establish that an IP message can be sent between all pairs of sites (X to Y, X to Z, Y to X, Y to Z, Z to X, Z to Y) with cost 5.

Note By default, all site links are transitive; that is, all site links for a specific transport implicitly belong to a single site link bridge for that transport. (For information about site link bridges, see "Site Link Bridges" later in this chapter.) If your IP network is not fully routed, you can turn off the transitive site link feature for the IP transport, in which case all IP site links are considered nontransitive and you can configure site link bridges.

The KCC determines the least-cost path from each site to every other site for each directory partition. The KCC then reviews the comparison of multiple paths to and from every destination and computes the spanning tree of the least-cost path.

For information about IP routing, see "Determining Network Connectivity Strategies" in the *Deployment Planning Guide*.

Replication Period

For each site link object, you can specify a value for the replication period, which determines how often replication occurs over the site link during the time that the schedule allows. For example, if the schedule allows replication between 02:00 hours and 04:00 hours, and the replication period is set for 30 minutes, replication can occur up to four times during the scheduled time.

The default replication period is 180 minutes, or 3 hours. When the KCC creates a connection between a domain controller in one site and a domain controller in another site, the replication period of the connection is the maximum period along the minimum-cost path of site link objects from one end of the connection to the other.

Schedule

In the case of RPC transport between sites, the replication between sites can be scheduled. Site links are associated with a schedule, which opens one or many windows for when replication can occur. If replication goes through multiple site links, there must be at least one common window; otherwise, the connection is treated as not available. For example, if site link 1 has a schedule (window of opportunity) of 18:00 hours to 24:00 hours and site link 2 has a schedule (window of opportunity) of 17:00 hours to 20:00 hours, the resulting window of opportunity is 18:00 hours through 20:00 hours, which is the intersection of site link 1 and site link 2.

Replication Path

The path that replication takes between sites is computed from the information on the site link objects. When a change is made to a site link setting, the following events must occur before the change takes effect:

- The site link change must replicate to each topology-generating system by using the previous topology.
- The KCC must run on the topology-generating systems.

As the path of connections is transitively figured through a set of site links, the attributes (settings) of the site link objects are "aggregated" along the path as follows:

- Costs are added together.
- The replication period is the maximum of the intervals that are set along the path.
- The options, if any are set, are "ANDed" together.

Note Options are the values of the *options* attribute on the site link object. The value of this attribute determines special behavior of the site link, such as reciprocal replication and intersite change notification. (For more information about these behaviors, see "Reciprocal Replication" and "Change Notification" later in this chapter.)

Thus the site link schedule is the "overlap" of all of the schedules of the subpaths. If none of the schedules overlap, the path is not used.

Site Link Settings and Replication Latency

The schedule determines the time intervals during which the site link is available, and the replication period determines how often replication can occur during those intervals. The interaction of these factors determines the replication latency. For sites where the maximum replication period within the site is 15 minutes, the worst-case, end-to-end replication latency from a source domain

controller to a destination domain controller in a remote site is the sum of the replication period settings for the connections between the source and destination sites, plus 15 minutes for each site in the path (including the source and destination sites). This sum assumes that the RPC transport is used between sites and that the required physical connections are available.

Interaction of Schedule and Replication Period

When multiple site links are required to complete replication for all sites, the replication periods on each link combine to affect the entire length of the connection between sites. In addition, when schedules on each link do not coincide, replication can occur only during the window of opportunity where the schedules intersect.

Suppose that site A and site B have site link AB, and site B and site C have site link BC. When a domain controller in site A replicates with a domain controller in site C, it can do so only as often as the maximum period set for site link AB and site link BC allow. Table 6.4 shows the site link settings that determine how often and during what times replication can occur between domain controllers in site A, site B, and site C.

Table 6.4 Replication Period and Schedule Settings for Two Site Links

Site Link	Replication Period	Schedule
AB	30 minutes	12:00 hours to 04:00 hours
BC	60 minutes	01:00 hours to 05:00 hours

Given the settings in Table 6.4, a domain controller in domain A can replicate with a domain controller in site B according to the AB site link schedule and period, which is once every 30 minutes between the hours of 12:00 and 04:00. However, assuming that there is no site link AC, a domain controller in site A can replicate with a domain controller in site C once every 60 minutes, which is the greater of the two replication periods, and between the hours of 01:00 and 04:00, which is where the schedules on the two site links intersect.

Schedule Implementation

The times that you can set in the **Schedule** setting on the site link are in one-hour increments. For example, you can schedule replication to occur between 00:00 hours and 01:00 hours, between 01:00 hours and 02:00 hours, and so forth. However, each block in the actual connection schedule is 15 minutes. For this reason, when you set a schedule of 01:00 hours to 02:00 hours, you can assume that replication is queued at some point between 01:00 hours and 01:14:59 hours.

Note RPC synchronous inbound replication is serialized so that if the server is busy replicating this directory partition from another source, replication from a different source does not begin until the first synchronization is complete. SMTP results are processed serially by order of arrival.

Specifically, a replication event is queued at time $t + n$, where t is the replication period that is applied across the schedule and n is a pseudo-random number between 1 minute and 15 minutes, inclusive. For example, if the site link indicates that replication can take place between 02:00 hours and 07:00 hours (inclusive), and the replication period is 2 hours (120 minutes), t is 02:00 hours, 04:00 hours, and 06:00 hours. A replication event is queued between 02:00 hours and 02:14:59 hours, and another replication event is queued between 04:00 hours and 04:14:59 hours. Assuming that the first replication event that was queued is complete, another replication event is queued between 06:00 hours and 06:14:59 hours. If the synchronization took longer than two hours, the second synchronization would be ignored because there is already an event in the queue.

Note The replication queue is shared with other events, and the time at which replication takes place is approximate. Duplicate replication events are not queued for the same directory partition and transport.

For information about how to create and configure site links, see Windows 2000 Server Help.

Preferred Bridgehead Server Configuration

Bridgehead servers must be able to accommodate more replication traffic than non-bridgehead servers, and you might want to choose which servers are to carry out this task. Knowing which system is acting as a bridgehead also can be useful for troubleshooting.

When you manually configure a single domain controller as the bridgehead server for a site, the KCC uses only that server. When multiple domain controllers in a site are configured to be preferred bridgehead servers, the KCC ultimately selects one of these servers on the basis on other variables.

Depending on what transports are available, which directory partitions must be replicated, and the availability of Global Catalog servers, multiple bridgehead servers might be required to replicate full copies of data from one site to another.

Suppose that there are two sites, site A and site B, and each site has a single domain controller from each of two domains, domain X and domain Y. In this case, the only way that replication of the respective domain directory partitions can occur between the two sites is if the domain controllers for domain X and domain Y are selected as bridgehead servers in each site. Therefore, if there is a single domain controller for a specific domain in a specific site, that domain controller must be a bridgehead server in its site because it can replicate domain data to only a domain controller in its own domain. In addition, that single domain controller must be able to connect to a bridgehead server in the alternative site that also holds the same domain directory partition.

If you want the KCC to consider certain domain controllers over others as bridgehead servers, you can specify a domain controller and an associated transport to indicate this preference by using the server object properties in Active Directory Sites and Services.

You specify the server when you select its server object, and then you add each transport for which the selected domain controller is a preferred bridgehead (IP for RPC over IP, or SMTP for SMTP over IP). If you select more than one server for a specific transport that can replicate a particular domain directory partition, the KCC chooses one arbitrarily. For information about how to specify preferred bridgehead servers, see Windows 2000 Server Help.

Bridgehead Server Failure

When a bridgehead server goes down, potentially it can cut off replication between this site and the other site. In most cases, the KCC selects a different bridgehead server automatically.

Failure of KCC-Selected Bridgehead Servers

Unless you specify a preferred bridgehead server or servers, the KCC selects them automatically. When the KCC selects bridgehead servers automatically and the current bridgehead server fails, after a time interval (the point at which a failure has occurred and the time since the last successful replication is greater than 2 hours), the KCC selects another bridgehead server to take its place. If all potential bridgehead servers are unavailable, the KCC logs an event that describes the condition.

Failure of Preferred Bridgehead Servers

If you explicitly set a preferred bridgehead server or servers and none is available, the KCC does not select alternative bridgehead servers automatically. In this case, the KCC logs an error message that states that you have designated preferred bridgehead servers that can replicate a specific directory partition, but none of them is available.

Replacement of a Failed Preferred Bridgehead Server

If you want the KCC to be able to fail over to other domain controllers but there are no other preferred bridgehead servers available, you must do one of the following *at a domain controller in each site*:

- Add new domain controllers as preferred bridgehead servers for the corresponding directory partitions, site, and transport. (If there is more than one domain represented in the site, you must add a preferred bridgehead server for the correct domain.)
- Remove all preferred bridgehead designations that you have made for the corresponding site and transport, in which case the KCC selects new ones automatically.

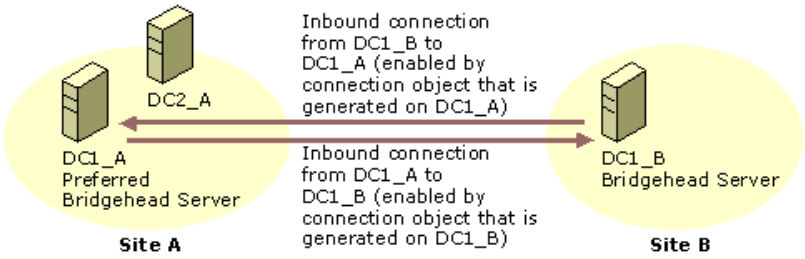
Important The KCC creates only inbound connections. A bridgehead server cannot create an outbound connection to another bridgehead server. For this reason, unless there is already a functioning replication path to the other site, changes to preferred bridgehead server status must be made on a domain controller in each affected site so that inbound connections are created in each site.

If the only preferred bridgehead server that is available for a specific directory partition and transport fails and you want to assign a new bridgehead server to replace it, you must add the new bridgehead server twice - once on a domain controller in the site of the failed bridgehead server, and once on a domain controller in the site on the other end of the affected site link. This process might involve two administrators if the site locations are far away from each other.

Note If there are preferred bridgehead servers available and you want to add another preferred bridgehead server in the site, you do not have to add the server in both sites because the change replicates to the other site through the currently available bridgehead servers.

If you remove all preferred bridgehead servers so that the KCC can select bridgehead servers automatically, you must remove them for each domain directory partition and for each transport on a domain controller in each affected site.

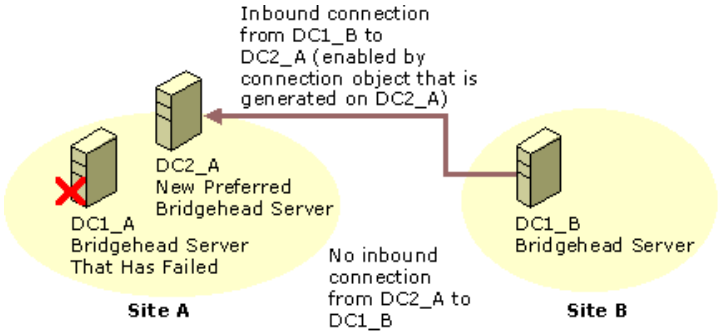
Figure 6.8 shows the connections between bridgehead servers in two sites. The bridgehead server in site A is a preferred bridgehead server.



If your browser does not support inline frames, click here to view on a separate page.

Figure 6.8 Two Sites and Two Bridgehead Servers with Inbound Connections from Each Other

If the preferred bridgehead server in site A fails, the bridgehead server in site B loses its inbound connection from the failed server and, thus, is disconnected from the site. If you assign a server to replace the failed bridgehead server in site A, the new bridgehead server creates only inbound connections. This change cannot replicate to site B because there is no inbound connection at the bridgehead server in site B. Figure 6.9 shows the connection that is created when you add a new bridgehead server in site A.



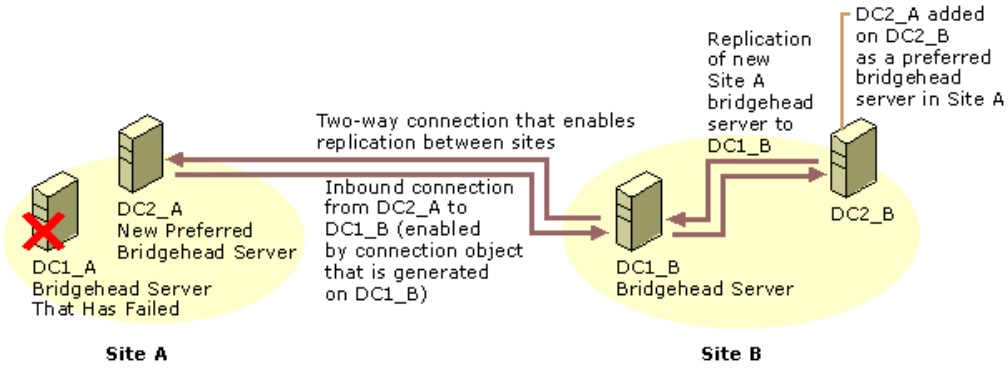
If your browser does not support inline frames, click here to view on a separate page.

Figure 6.9 New Inbound Connection from the Existing Server with No Inbound Connection from the New Bridgehead Server

Until you add the new site A bridgehead server to a domain controller in site B, there is no inbound connection possible from site A to site B, even though you have added the bridgehead server in site A. The reason is that the KCC creates only inbound connections and has no knowledge of the new server.

In Figure 6.10, the new bridgehead server for site A has been added to a domain controller in site B. The figure shows the new inbound connection that results. The sequence of events is as follows:

1. An administrator in site B goes to a domain controller and adds DC2_A as a preferred bridgehead server by selecting DC2_A in the Servers container under site A in Active Directory Sites and Services. He or she then adds the server to the list of preferred bridgehead servers for the appropriate transport.
2. The bridgehead server designation is replicated (as a change to the configuration directory partition) to all domain controllers in site B, including the bridgehead server (DC1_B).
3. DC1_B generates an inbound connection object from DC2_A, which completes the two-way replication route between bridgehead servers in site A and site B. At this point, replication is possible between the two bridgehead servers.



If your browser does not support inline frames, click here to view on a separate page.

Figure 6.10 Two-Way Connection Between Sites After Adding the New Site A Bridgehead Server in Site B

Failure of a Preferred Bridgehead as a Result of Incorrect Configuration

If you have configured preferred bridgehead servers but none of them is capable of replicating a directory partition that must be replicated, the KCC logs an event for a configuration error. (A domain that has servers in the site is not represented by a bridgehead server.) The KCC then proceeds to select an alternative in the same manner as if no preferred bridgehead servers are configured.

Site Link Bridges

A site link bridge object represents a set of site links, all of whose sites can communicate through some transport. A site link bridge usually corresponds to a router (or a set of routers) in an IP network.

Note If no bridgehead server that is capable of the site link bridge transport is available in two linked sites, a route is not available.

By default, all site links that you create are bridged ("transitive"); all site links for a specific transport implicitly belong to a single site link bridge for that transport. Therefore, in the common case of a fully routed IP network, you do not need to configure any site link bridges. Figure 6.11 shows a case where three sites are connected by two site links and the site link bridge allows connections to be created between two sites that are not connected by an explicit site link.

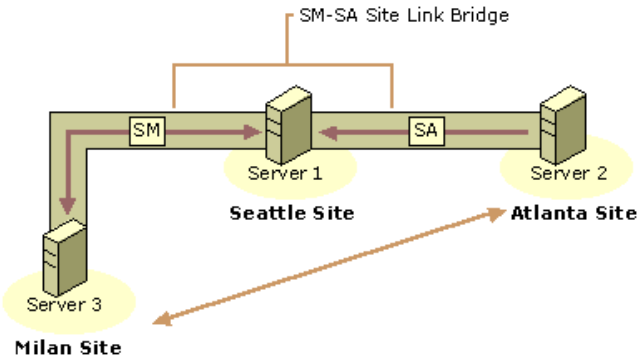


Figure 6.11 Site Link Bridge That Enables Connections Between Milan and Atlanta Sites

If your IP network is not fully routed, you can turn off the **Bridge all site links** for IP transport (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 nontransitive, and you can configure site link bridges to model the actual routing behavior of your network.

Managing Site Link Bridges

You create a site link bridge object for a specific transport by specifying two or more site links for the specified transport.

To understand what a site link bridge means, consider the following example:

- Site link SM connects the Seattle site and the Milan site over IP with cost 4.
- Site link SA connects the Seattle site and the Atlanta site over IP with cost 3.
- There is no site link between the Milan site and the Atlanta site.
- Site link bridge SM-SA connects site link SM and site link SA.

In this simple example, the site link bridge SM-SA implies that an IP message can be sent from the Milan site to the Atlanta site with cost 4+3 = 7.

Each site link in a bridge must have at least one site in common with another site link in the bridge. Otherwise, the bridge cannot compute the cost from sites in one link to the sites in other links of the bridge. For example, if you have four sites (W, X, Y, and Z), a site link WX that connects W and X, and a site link YZ that connects Y and Z, a site link bridge that connects WX and YZ serves no purpose.

Separate site link bridges, even for the same transport, are independent. To illustrate this independence, the following objects are added to the Milan-Seattle-Atlanta example:

- Site link DA connects the Detroit site and the Atlanta site over IP with cost 2.
- Site link bridge DA-SA connects site link DA and site link SA.

The presence of this additional bridge means that an IP message can be sent from the Seattle site to the Detroit site with cost 2 + 3 = 5; but it does not imply that an IP message can be sent from the Detroit site to the Milan site with cost 2 + 3 + 4 = 9. In almost all cases, you use a single site link bridge to model the entire IP network.

Performance Considerations

Any network that you can describe by a combination of site links and site link bridges also can be described by site links alone. The advantage to bridging all site links is that your network description is much smaller and easier to maintain because you don't need a

site link to describe every possible path between pairs of sites.

However, when the number of sites exceeds 200, periods of high CPU activity occur every 15 minutes when the KCC runs. The **Bridge all site links** setting creates a single bridge for the entire network, which generates more routes that must be processed than if site link bridges are not used or are applied selectively.

For example, under the conditions identified in Figure 6.11, the following factors affect KCC performance:

- The KCC recognizes the explicit site links between Atlanta and Seattle and between Seattle and Milan.
- With the site link bridge in place, the KCC also must consider the implicit paths between Milan and Atlanta as a single path with a combined cost.
- Where the site links represent hops between domain controllers in the same domain, replication changes flow transitively without the site link bridge. When a site link bridge is in place, the KCC must compute transitivity between the sites.
- The site link bridge adds more connection possibilities, which are usually eliminated because they have a higher aggregated cost. The KCC must spend extra time and cycles to eliminate these connection possibilities.

Thus, in a large network where processing time is a concern, there are performance advantages to turning off **Bridge all site links** and configuring site link bridges only where they are advantageous. If you still experience delays, the next step is to replace the bridges with a large number of sites that have explicit site links. For more information about KCC scaling recommendations, see the Microsoft Knowledge Base link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources . Search the Knowledge Base by using the keyword "Q244368."

Global Catalog Replication

A Global Catalog server is a domain controller that stores specific information about all objects in the forest. The Global Catalog is required for the logon process, so it is best to have at least one Global Catalog server per site. The Global Catalog stores a replica of every directory partition in the forest: It stores full replicas of the schema and configuration directory partitions, a full replica of the domain directory partition for which the domain controller is authoritative, and partial 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 from the domain directory partition to the corresponding directory partition replicas on all authoritative domain controllers and also to all Global Catalog servers.

Figure 6.12 depicts logical directory partitions in the Active Directory database of a Global Catalog server. (The database itself, Ntds.dit, is not actually partitioned.) The top three segments represent directory partitions that are full replicas for the domain controller. The bottom three segments represent directory partitions that are partial replicas for the Global Catalog.

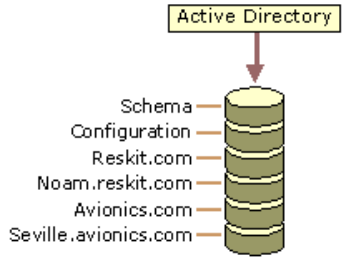
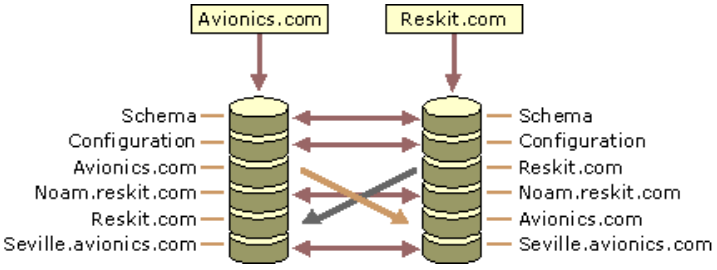


Figure 6.12 Directory Database on a Domain Controller That Is a Global Catalog

Global Catalog servers request updates from a source domain controller for each domain directory partition in the forest. This source domain controller can be either a normal domain controller or another Global Catalog server.

Figure 6.13 shows the partner associations between directory partitions in two Global Catalog servers that are authoritative domain controllers for different domains. As is true for all domain controllers, the Global Catalog uses a single topology to replicate the schema and configuration directory partitions, and it uses a separate topology for each domain directory partition. Replication is two-way between the domain directory partitions.

One server is authoritative for avionics.com; the other server is authoritative for reskit.com. As such, the avionics.com domain controller can be the source for replication to the partial replica of avionics.com on the reskit.com Global Catalog server, and the reskit.com domain controller can be the source for replication to the partial replica of reskit.com on the avionics.com Global Catalog server. The connection arrows indicate the one-way flow of replication from the read-write sources to the read-only destinations. In the case of the noam.reskit.com domain, neither domain controller is authoritative for that domain, so the Global Catalog servers replicate these partial replicas to and from each other.



If your browser does not support inline frames, click here to view on a separate page.

Figure 6.13 Directory Partition Connections Between Two Global Catalog Servers in Different Domains

Replication of Removed Properties

When you want to remove an attribute from the Global Catalog, you must set its *isMemberOfPartialAttributeSet* value to FALSE. The attribute then is removed from the Global Catalog immediately after the next replication cycle.

Added Attributes

If an attribute has been added to the partial attribute set, the domain controller must replicate the value of this attribute for each of its partial directory partition replica objects. This is accomplished by performing a full synchronization across all of the Global Catalog's replication connections. If the partial directory partition replica can be synchronized over an RPC connection, the domain controller attempts a full synchronization over an RPC connection before it uses any other connections; if full synchronization is completed, the up-to-dateness vector that it creates optimizes later full synchronization on other connections.

Caution Caution should be exercised when you add attributes to the Global Catalog attribute set because doing so causes full synchronization of the Global Catalog on all Global Catalog servers in the forest. 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.

Universal Group Replication

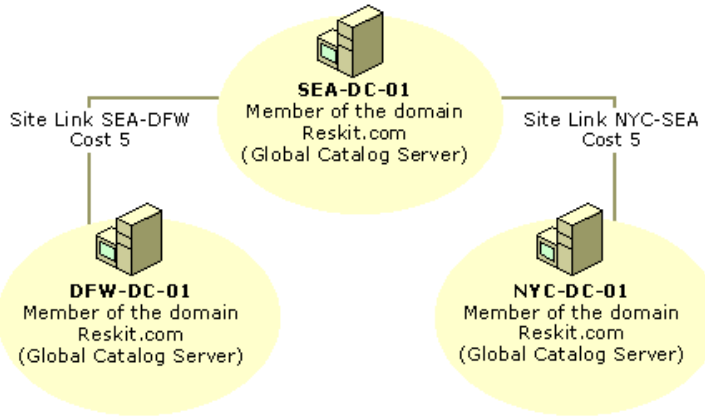
A *universal group* can have members from any domain in the forest, and thus the membership for universal groups cannot be stored on every domain controller (each domain controller stores objects for only one domain), but only on Global Catalog servers, which store every object in the forest. Therefore, the Global Catalog servers are the only domain controllers that can enumerate the membership of a universal group. For this reason, a Global Catalog server is required for logging on to domains that use universal groups. During the logon process, the Global Catalog enumerates the membership of universal groups and attaches any found membership to the security token of the user. Other types of groups (global and domain local) are not enumerated by the Global Catalog; only the group object name is listed. The enumeration of global and domain local groups is the responsibility of the resource domain controller. For replication, this arrangement means that the replication of global and domain local group memberships is not required by Global Catalog servers, which significantly reduces replication traffic.

Scenarios for Replication Between Sites

The following scenarios combine different approaches to using site links and site link bridges.

One Domain Spanning Multiple Sites

In Figure 6.14, two site links have been defined, NYC-SEA and SEA-DFW. In this environment, the default behavior of transitive site links has been disabled because the network is not fully routed, so there is no site link bridge. For this reason, the KCC can create connections only between NYC and SEA and between SEA and DFW. Because all domain controllers are in the same domain and therefore maintain a full copy of the same domain directory partition, replication can occur through the hub site SEA without requiring a connection between DFW and NYC. As a change occurs in NYC, this change (in its entirety) is replicated to SEA. Because domain controllers that are holding full domain directory partition replicas can replicate inbound changes from only another full domain replica, SEA-DC-01 is a valid replication partner for DFW-DC-01, and therefore, in turn, DFW-DC-01 transitively replicates the NYC change from SEA-DC-01. The reverse is also true - changes replicate from DFW to SEA and then to NYC. Thus, changes that originate in SEA replicate to both DFW and NYC directly, but replication must occur twice for changes to flow from DFW to NYC and from NYC to DFW.



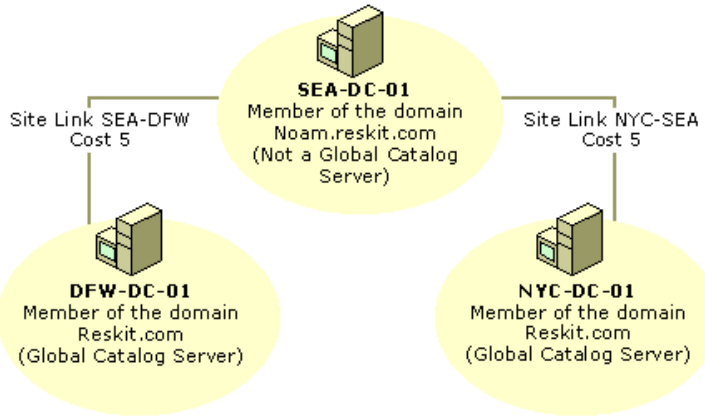
If your browser does not support inline frames, click here to view on a separate page.

Figure 6.14 A Single Domain in Three Sites Connected by Nontransitive Site Links

Site Links with Two Domains in Three Sites

Figure 6.15 shows that the SEA site contains a domain controller for the noam.reskit.com domain but not for the reskit.com domain. In this case, how would changes replicate from NYC to DFW?

If transitive site links were enabled or if a site link bridge were created manually between NYC-SEA and SEA-DFW, the KCC could create a connection object to replicate data between DFW and NYC. If neither condition were true, DFW-DC-01 would never see changes from NYC, or vice versa, because the only domain controller in the SEA site is a domain controller for a different domain (noam.reskit.com). If SEA-DC-01 does not contain a replica of the reskit.com domain directory partition, it cannot transitively replicate domain data between NYC and DFW. Schema and configuration data can replicate between NYC and SEA and between SEA and DFW.



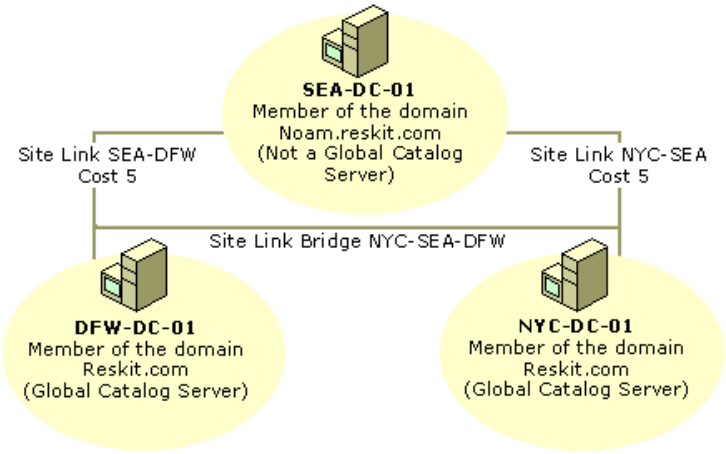
If your browser does not support inline frames, click here to view on a separate page.

Figure 6.15 Site Links Between Three Sites for Two Domains

In Figure 6.15, if SEA-DC-01 were to become a Global Catalog server, the scenario would not change relative to replication because the domain controller is authoritative for the noam.reskit.com domain and, as such, would hold a full domain directory partition for only that domain. As a Global Catalog server, SEA-DC-01 would hold only a partial replica of the reskit.com domain. In this scenario, SEA-DC-01 would receive changes from NYC, but DFW-DC-01 would not be able to replicate from SEA-DC-01 because SEA-DC-01 contains only a partial replica of the reskit.com partition.

Site Link Bridge with Two Domains in Different Sites

In Figure 6.16, with transitive site links disabled, adding a site link bridge that contains DFW-SEA and SEA-NYC changes the rules on the possible replication partners that the KCC can select. The bridge gives the KCC a route from DFW to NYC in which to create a connection object that allows replication to occur, even if no domain controller for the same domain directory partition exists in SEA. Schema and configuration changes can be replicated over this connection.



If your browser does not support inline frames, click here to view on a separate page.

Figure 6.16 Site Link Bridge That Connects SEA-DFW and SEA-NYC

By default, all site links are defined as transitive and do not require definitions of site link bridges. Without a site link bridge, the KCC still constructs connection objects between sites such that replication is the most efficient, based on the defined costs for each site link. However, with a site link bridge, the connection objects might reflect a direct connection from a domain controller in one site to a domain controller in another site where there is no site link.

For information about network routing, see "Unicast IP Routing" in the *Microsoft® Windows® 2000 Server Resource Kit Internetworking Guide* and see "Determining Network Connectivity Strategies" in the *Deployment Planning Guide*.

KCC and Topology Generation

The KCC is a built-in process that runs on all domain controllers. It is a dynamic-link library that modifies data in the local directory in response to systemwide changes, which are made known to the KCC by changes to the data within Active Directory. The KCC generates and maintains the replication topology for replication within sites and between sites.

The KCC has two major functions:

- Configures replication connections (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 KCC per site generates all connections between sites.
- Converts the connection objects that represent inbound replication to the local domain controller into the replication agreements 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

Although the work done by the KCC is evidenced by the automatically generated connection objects that are visible in Active Directory Sites and Services, there is no UI component for managing the KCC per se.

Most replication tasks that affect the KCC can be managed by using Active Directory Sites and Services. For information about non-MMC tools that can be used for advanced replication management and diagnosis, see "Active Directory Diagnostics, Troubleshooting, and Recovery" in this book.

Active Directory Sites and Services

Active Directory Sites and Services is the primary administrative tool that is used to manage replication. Use this tool 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 intervention. Replication between sites is managed most effectively by changing the settings on the site link objects, as described in this chapter.

For information about procedures for managing replication through Active Directory Sites and Services, see Windows 2000 Server Help.

Event Viewer

Communication from the KCC to the administrator occurs through event logs that you can view in Event Viewer.

The following examples contain a few of the events that are generated by the KCC in the event log:

- Event 1009 (informational): The consistency checker has started updating the replication topology for this server.
- Event 1013 (informational): The replication topology update task terminated normally.
- Event 1265 (warning): The attempt to establish a replication link with parameters <*parameters*> failed with the following status: <*error message*>. The record data is the status code. This operation is going to be re-tried.

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 HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \NTDS \Diagnostics in the registry. Increasing the level of detail can be

used to better understand the behavior of the KCC in different situations. However, a logging level value of 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. For information about how to modify the registry to increase logging levels, see "Active Directory Diagnostics, Troubleshooting, and Recovery" in this book. For information about using Event Viewer, see Windows 2000 Server Help.

Objects Required by the KCC for Building Topology

When the KCC builds the topology, it must determine which servers that 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, which identifies the domain controller as having Active Directory installed. 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.

Topology Generation Phases

There are two phases of topology generation. During phase one, the KCC evaluates the current topology, determines whether replication failures have occurred with the existing connections, and constructs whatever new connection objects are required to complete the replication topology. During phase two, the KCC implements, or "translates," the decisions that were made in phase one into agreements between the replication partners.

Phase One: Evaluating the Current Topology and Generating Connection Objects

The topology is evaluated by reading the connection objects. When the KCC notices a connection object, it reads the NTDS Settings object of the source domain controller (indicated by the *fromServer* value on the connection object) to determine what directory partitions its destination controller has in common with the source domain controller. The *hasMasterNCs* attribute (where "NC" stands for "naming context," a synonym for "directory partition") of an NTDS Settings object contains the set of writable (non-Global Catalog) directory partitions that are located on that domain controller. The *hasPartialReplicaNCs* attribute contains the set of partial-replica directory partitions (Global Catalog partitions) that are located on that domain controller. For each directory partition that the two domain controllers have in common and that matches the full and partial characteristics of a replication source, the KCC creates (or updates) a replication agreement.

Note Within a site, all KCCs generate connection objects for replication within the site. When there is more than one site, a single KCC in each site generates all connection objects for replication between sites.

Phase Two: Translating Connections

In phase two, all KCCs process their connection objects and translate them into connection agreements between pairs of domain controllers. At specified intervals, Active Directory replicates data from one replication partner to the other for directory partitions that they have in common. These replication agreements do not appear in the administrative tools. They are used internally by the replication engine to track the directory partitions that are to be replicated from specified servers.

For example, suppose that you define a connection object between two domain controllers from different domains. In phase two, assuming that neither of these domain controllers is a Global Catalog server, the KCC identifies the only two directory partitions that the domain controllers have in common - the schema directory partition and the configuration directory partition. If you create a connection object that links domain controllers in the same domain, at least three directory partitions are replicated: the schema directory partition, the configuration directory partition, and the domain directory partition.

In contrast, if the connection object is established between two domain controllers that are Global Catalog servers, a partial replica of each directory partition (which includes only specified attributes) in the forest is replicated between the two domain controllers.

Intervals at Which the KCC Runs

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. This interval can be modified by changing the **Repl topology update delay (secs)** entry in HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \NTDS \Parameters as follows:

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 **Repl topology update period (secs)** entry in HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \NTDS \Parameters as follows:

Value: Number of seconds between KCC topology updates

Default: 900 seconds (15 minutes)

Data type: REG_DWORD

Caution Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. There are programs available in Control Panel or Microsoft Management Console (MMC) for performing most administrative tasks. These programs provide safeguards that prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Registry editors bypass the standard safeguards that are provided by these administrative tools. Modifying the registry is recommended only when no administrative tool is available. Before you make changes to the registry, it is recommended that you back up any valuable data on the computer. For instructions about how to edit registry entries, see Help for the registry editor that you are using. For more information about the registry, see the *Microsoft Windows 2000 Resource Kit* Technical Reference to the Windows 2000 Registry (Regentry.chm).

Automated Replication Topology Generation Within a Site

During topology creation within a site, KCC activity includes topology translation and topology generation. The KCC "translates" the information provided by an algorithm and then creates (generates) connection objects that implement the topology that is dictated by the translation. In general, topology translation and generation are accomplished as follows:

- The KCC on each domain controller runs an algorithm to determine the topology for replication within its site. All servers in the same site have the same information about each other, so they all deduce the same topology.

- The KCC creates the connection objects for servers from which the domain controller receives data (source servers), as dictated by the algorithmically generated topology.
- The sum total of the connection objects for all servers is the desired topology.

By listing and sorting the domain controllers that hold replicas of the same partition, the topology is generated and then optimized to minimize the number of hops.

Forced Topology Generation

Topology generation creates the topology for replication within a site automatically. Topology generation occurs on a schedule that determines how often the KCC runs. Topology generation can also be started manually by right-clicking the NTDS Settings object, clicking **All Tasks**, and then clicking **Check Replication Topology**.

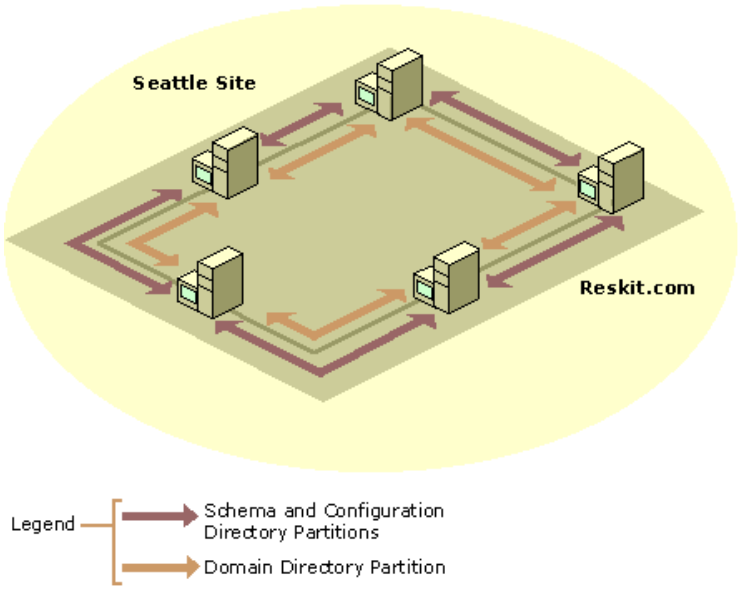
Simplified Ring Topology Generation

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.
- 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.

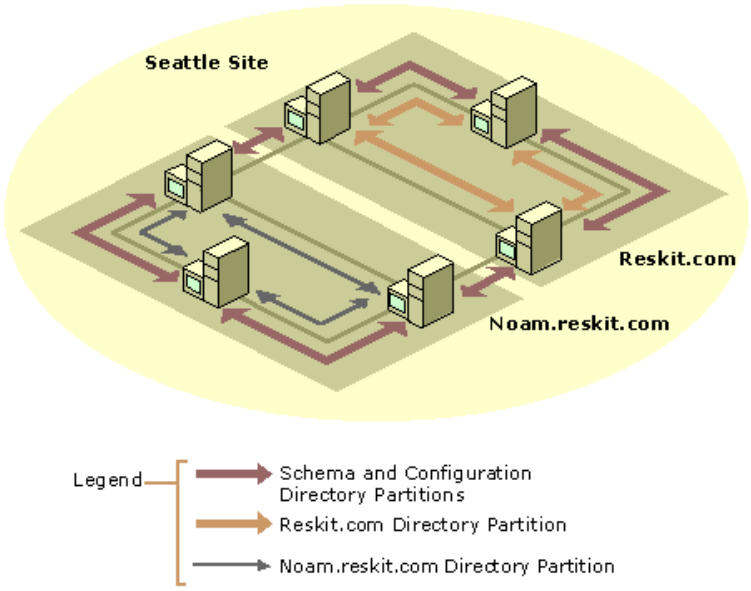
The simplest case scenario - seven or fewer domain controllers, all in the same domain and site - would result in the topology shown in Figure 6.17. Even if one or all of these domain controllers were Global Catalog servers, when the KCC runs on those particular computers, no extra connections would 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.



If your browser does not support inline frames, click here to view on a separate page.

Figure 6.17 Simple Ring Topology That Requires No Optimization

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. Figure 6.18 illustrates the topology for domain controllers from two domains in the same site with no Global Catalog servers defined in the site.



If your browser does not support inline frames, click here to view on a separate page.

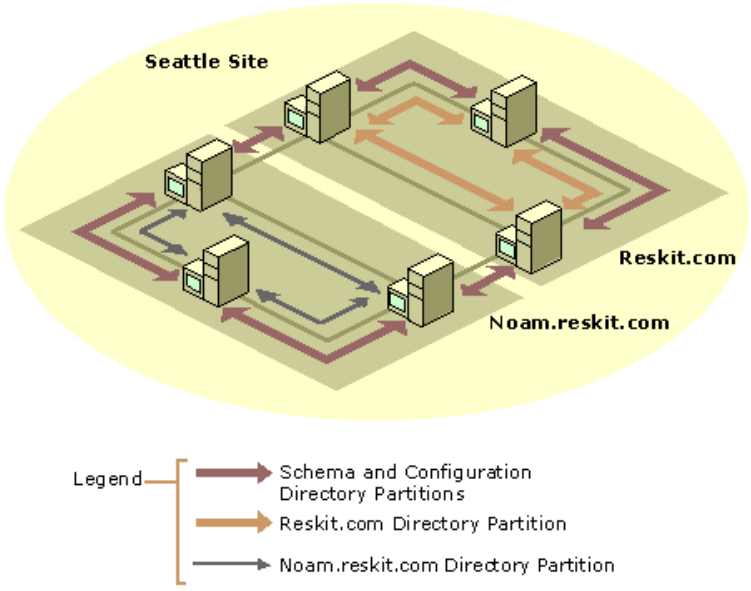
Figure 6.18 Ring Topology for Two Domains in a Site That Has No Global Catalog Server

Note The examples in Figure 6.18 and Figure 6.19 are designed to illustrate only how the KCC creates connections, not how to configure a site. A Global Catalog server is a requirement for logging on to a domain, and for this reason, it is advisable to have at least one Global Catalog server in a site. If a Global Catalog server is not available in a site and there is a Global Catalog server in a remote site, the server in the remote site can be used for the logon process. If no Global Catalog is available in any site, the logon process proceeds with cached logon information. (For more information about Global Catalog support for logging on to domains, see "Active Directory Logical Structure" in this book.)

Expanded Ring Topology

When the number of servers grows beyond seven, 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 to ensure that no domain controller is more than three replication hops from another domain controller (that is, a change takes no more than three hops before it reaches another domain controller that has already received the change by another path). These optimizing connections are created at random and are not necessarily created on every third domain controller.

In Figure 6.19, 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 optimizing connections. Although they are located in the same domain and site, Domain Controller A and Domain Controller B are more than three hops away from each other. The optimizing connections for the domain, schema, and configuration directory partitions that might be created from Domain Controller A to Domain Controller B 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.



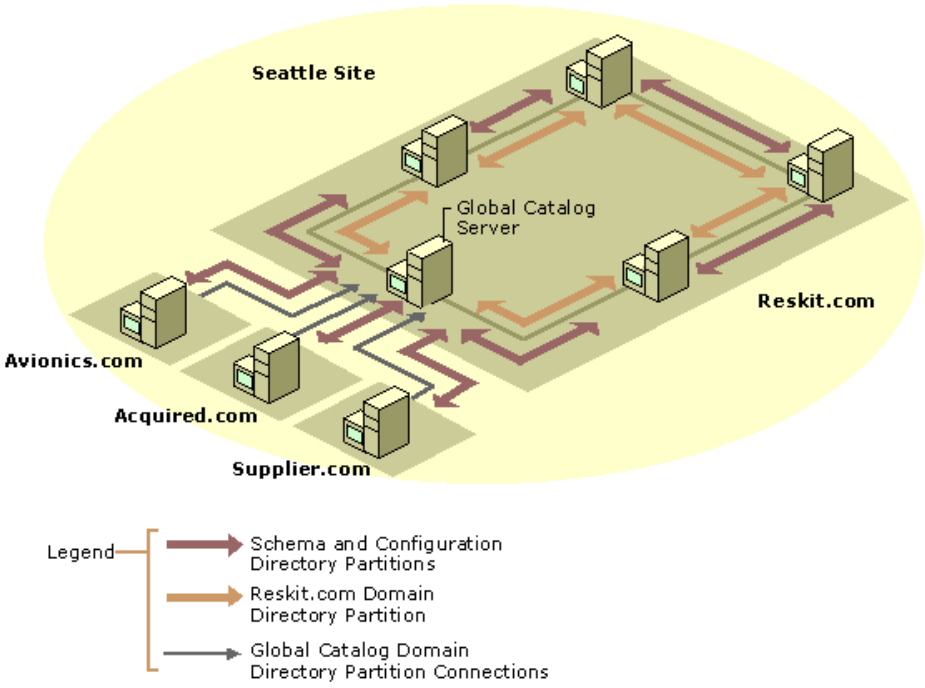
If your browser does not support inline frames, click here to view on a separate page.

Figure 6.19 Optimized Connections for Ten Domain Controllers in the Same Domain in a Single Site

Figure 6.20 illustrates the connections required between a Global Catalog domain controller and three other domains to which it does not belong. When a Global Catalog server is added to the site, additional connections are required to replicate copies of the directory partitions to the domains to which the Global Catalog server does not belong. Because the reskit.com domain has only seven servers, no optimizing connections are required in the replication topology for the reskit.com directory partition. However, the Global Catalog

server is the source for each domain directory partition in the forest, and in this example, the KCC has created connection objects to replicate from domain controllers for each of those domain directory partitions within the site. Wherever a domain directory partition is replicated, the KCC also uses the connection to replicate the schema and configuration directory partitions.

Note Connection objects are generated independently for the configuration and schema directory partitions (one connection) and for the separate domain directory partitions, unless a connection from the same source to destination domain controllers already exists for one directory partition, in which case the same connection is used for both (a duplicate connection is not created).



If your browser does not support inline frames, click here to view on a separate page.

Figure 6.20 Optimized Connections That Are Required for Site That Has Four Domains and a Global Catalog Server

Optimized Ring Topology Connections Within a Site

Connections are added to optimize a ring topology within a site on the basis of the answer to the following question:

Given a set of nodes in a ring, what is the minimum number of connections, *n*, that each server must have to ensure a path of no more than three hops to another server?

Given *n*, topology generation proceeds as follows.

If the local server does not have *n* extra connections, do the following:

- Choose *n* other servers randomly in the site as source servers.
- For each of those servers, create a connection object.

This approach approximates the minimum-hop goal of three servers. In addition, it grows well, because as the site grows in server count, old optimizing connections are still useful and are not removed. Also, every time an additional 9 to 11 servers are added, a connection object is deleted at random; then a new one is created, ideally having one of the new servers as its source. This process ensures that, over time, the extra edges are distributed well over the entire site.

Excluded Nonresponding Servers

The KCC automatically rebuilds the replication topology when it recognizes that a domain controller has failed or is unresponsive.

The criteria that the KCC uses to determine when a domain controller is not responsive depend upon whether the server computer is within the site or not. Two thresholds must be reached before a domain controller is declared "unavailable" by the KCC:

- The requesting domain controller must have made *n* number of attempts to replicate from the target domain controller.
  - For replication between sites, the default value is 1 attempt.
  - For replication within a site, the following distinctions are made between the two immediate neighbors (in the ring) and the optimizing connections:

  For immediate neighbors, the default value is 0 failed attempts. (Thus, as soon as an attempt fails, a new server is tried.)

  For optimizing connections, the default value is 1 failed attempt. (Thus, as soon as a second failed attempt occurs, a new server is tried.)
- A certain amount of time must pass since the last successful replication attempt.
  - For replication between sites, the default time is 2 hours.
  - For replication within a site, a distinction is made between the two immediate neighbors (in the ring) and the optimizing connections:

  For immediate neighbors, the default time is 2 hours.

  For optimizing connections, the default value is 12 hours.

To modify the thresholds for excluding nonresponding servers, use the following registry entries in HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \NTDS \Parameters, with the data type REG_DWORD. You can modify these values to any desired value as follows:

For replication between sites, use the following entries:

- IntersiteFailuresAllowed

  Value: Number of failed attempts

Default: 1

- MaxFailureTimeForIntersiteLink (secs)

  Value: Time that must elapse before being considered stale, in seconds

  Default: 7200 (2 hours)

For optimizing connections within a site, use the following entries:

- NonCriticalLinkFailuresAllowed

  Value: Number of failed attempts

  Default: 1

- MaxFailureTimeForNonCriticalLink

  Value: Time that must elapse before considered stale, in seconds

  Default: 43200 (12 hours)

For immediate neighbor connections within a site, use the following entries:

- CriticalLinkFailuresAllowed

  Value: Number of failed attempts

  Default: 0

- MaxFailureTimeForCriticalLink

  Value: Time that must elapse before considered stale, in seconds

  Default: 7200 (2 hours)

Caution Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. There are programs available in Control Panel or Microsoft Management Console (MMC) for performing most administrative tasks. These programs provide safeguards that prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Registry editors bypass the standard safeguards that are provided by these administrative tools. Modifying the registry is recommended only when no administrative tool is available. Before you make changes to the registry, it is recommended that you back up any valuable data on the computer. For instructions about how to edit registry entries, see Help for the registry editor that you are using. For more information about the registry, see the *Microsoft Windows 2000 Resource Kit* Technical Reference to the Windows 2000 Registry (Regentry.chm).

When the original domain controller begins responding again, the KCC automatically restores the replication topology to its pre-failure condition the next time that the KCC runs.

Fully Optimized Ring Topology Generation

Taking the addition of extra connections, management of nonresponding servers, and growth-management mechanisms into account, the fully optimized intrasite topology generation proceeds and the appropriate connection objects are created and deleted according to the available criteria.

Note Connection objects from nonresponding servers are not deleted because the condition is expected to be transient.

The following key points characterize the optimized approach to topology generation:

- Sorting the servers ensures that each server arrives at the same desired topology.
- Intrasite topology generation never creates or deletes connection objects that are not directly under the NTDS Settings object for the local domain controller.

Automated Intersite Topology Generation

Topology generation for replication between sites is more complex than for replication within a site because Windows 2000 supports replication between sites over asynchronous transport (SMTP). Therefore, whereas for intrasite topology generation the KCC can assume that any server can replicate to any other server, the same assumption cannot be made for replication between sites.

Intersite Topology Generator Role

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. As part of this effort, one domain controller per site assumes the role of the intersite topology generator. The KCC on this domain controller is responsible for creating the connections between the domain controllers in its site and the domain controllers in other sites, which includes specifically the inbound replication connection objects for all bridgehead servers in the site in which the domain controller is located.

After the intersite topology generator assesses the topology and determines that its own site is the only site, it performs no further processing because no connections between sites are possible for the current configuration.

Generation of Connections Between Sites vs. Generation of Connections Within a Site

Connection objects between bridgehead servers for replication between sites and connection objects for connections within a site are created differently.

When the KCC on each domain controller generates the intrasite topology for the site in which it resides, the KCC creates a connection object in Active Directory only when a connection object is required for the local computer. This change propagates to other domain controllers through the normal replication process. Each domain controller uses the same algorithm to compute the replication topology; in a state of equilibrium between domain controllers, each domain controller arrives at the same result with respect to what the replication topology should be. In the process, each domain controller creates its own connection objects.

For connections between sites, in each site, the KCC that has the intersite topology generator role (regardless of the domain) is responsible for reviewing the intersite topology and creating inbound replication connection objects as necessary for bridgehead servers in the site in which it resides.

When the intersite topology generator determines that a connection object needs to be modified on a specific bridgehead server in the site, the intersite topology generator makes the change to its local Active Directory copy. These changes propagate to the bridgehead servers in the site as part of normal replication within the site. When the KCC on the bridgehead server reviews the topology after receiving these changes, it translates the connection objects into replication agreements (replication partners) that Active Directory uses to replicate data from remote bridgehead servers.

Bridgehead Server Selection

As the KCC constructs the intersite topology for each directory partition, the servers in each site are evaluated for becoming bridgehead

servers. If preferred bridgehead servers are configured, these servers are the candidates for selection. Otherwise, all domain controllers in the site that host the directory partition and can communicate over a specific transport are candidates for becoming a bridgehead server. In either case, the first domain controller that meets the requirements becomes the bridgehead server. All preferred bridgehead servers in the same site that are configured for the same transport are considered to be equal. In a state of equilibrium of the configuration directory partition, the intersite topology generator from each site selects the same bridgehead server, given the same requirements. For example, if the same bridgehead server (either preferred or automatically selected) is capable of replication over the IP transport and holds the requested directory partition, that server is selected by bridgehead servers from other sites that require the same directory partition data.

If preferred bridgehead servers are defined for a specific site and all of the servers specified are unavailable, no failover is performed to the other domain controllers in the site, even if they can act as bridgehead servers. The same is true in the case where no preferred bridgeheads are specified and where all domain controllers that can act as a bridgehead for the required data are unavailable.

Intersite Topology Generator Role Owner

The current owner of the intersite topology generator role is communicated through the normal Active Directory replication process. Initially, the first domain controller in the site becomes the intersite topology generator for the site. The role does not change as additional domain controllers are added to the site until the current intersite topology generator becomes unavailable.

To determine the intersite topology generator role owner for a site

1. In Active Directory Sites and Services, click the site object.
2. In the details pane, right-click the **NTDS Site Settings** object, and then click **Properties**. The current role owner appears in the **Server** box under **Inter-Site Topology Generator**.

Note You cannot change the intersite topology generator role.

Intersite Topology Generator Role Owner Notification

At 30-minute intervals, the current intersite topology generator notifies every other domain controller in the site of its existence by writing the attribute *interSiteTopologyGenerator* on the NTDS Settings object under its domain controller object in the configuration directory partition.

As the *interSiteTopologyGenerator* attribute gets propagated to other domain controllers by Active Directory replication, the KCC on each of these computers monitors this attribute to verify that it has been written. If a period of 60 minutes elapses without a modification, a new intersite topology generator takes over.

Establishing a New Intersite Topology Generator

When a new intersite topology generator is required, each domain controller requests the list of servers in the site in ascending order. The domain controller that takes over the role is the one that is next in the list of servers after the current owner. This domain controller then writes the *interSiteTopologyGenerator* attribute and performs the necessary KCC processes to manage inbound connection objects for bridgehead servers.

When there are two domain controllers in the site that appear to own the intersite topology generator role, there might be a temporary state of inbound replication connection objects being created by both computers. However, after replication occurs and all domain controllers receive the change that identifies the new intersite topology generator, the KCC on the intersite topology generator adjusts the topology.

Intersite Topology Generator and Modified Connections

It is possible for a connection object to be modified by an administrator on one domain controller and be modified subsequently by the KCC on another domain controller prior to the initial change being replicated. Overwriting such a change can occur within the local site or when a connection object changes in a remote site. By default, the KCC runs every 15 minutes. If the connection object change is not replicated to another domain controller before the KCC on that domain controller runs, the KCC on that domain controller might modify the same connection object. In this case, ownership of the connection object belongs to the KCC because the latest write to the connection object is the write that is applied.

To ensure that modification of an intersite connection object is not overwritten by the KCC, make the modification on the computer that has the intersite topology generator role in the site of the modified connection object.

Security Between Replication Partners

Security of replication is important to ensure that an unauthorized program cannot act as a replication destination because replication destinations have access to secrets. Security must also prevent an unauthorized program from acting as a replication source because a rogue replication source can pretend to originate updates and thereby make unauthorized directory changes. Therefore, mutual authentication of replication partners and access control at the replication source are both required. The security technology that is used in replication depends on the replication transport.

RPC Transport Security

For authentication, mutual authentication is performed either by the Kerberos v5 authentication protocol or by NTLM when authenticating to a Microsoft® Windows® NT version 4.0-based backup domain controller.

For authorization, the DS-Replication-Get-Changes control access right provides access control at the replication source. This control access right is granted in access control entries (ACEs) in the security descriptor on the topmost object in the directory partition. By default, this right is granted to Enterprise Domain Controllers and Domain Admins and is required in order to run the Active Directory Installation Wizard (Dcpromo.exe).

For information about Active Directory authentication, see "Authentication" in this book. For information about Active Directory authorization, see "Access Control" in this book. For information about the Active Directory Installation wizard, see "Active Directory Data Storage" in this book.

ISM Transport Security

The authentication and encryption mechanism that ISM uses is similar to that of Secure/Multipurpose Internet Mail Extensions (S/MIME), which is a standard that enables binary data to be published and read on the Internet. By using S/MIME, the sender always includes its own certificate in addition to the certificate-authority certificate chain that extends to the root certifying authority. Requests to the replication source are signed but not sealed. The request does not contain any secret data, so the domain controller does not require knowledge of the certificates for any other domain controllers. Responses to the replication destination are signed and sealed by using the local domain controller certificate plus the certificate that the requester included in its request message.

Authentication is possible because all certificates used for Active Directory replication (that is, "DomainController" certificates) contain an attribute (*altSubjectName*) that identifies the *objectGuid* attribute value of the owner's computer object. The mapping of the computer *objectGuid* to the server object can be done by any domain controller because the required information is stored in the configuration directory partition.

For authorization, the receiving domain controller verifies the validity of the certificate (including verifying the fact that it was issued by

a trusted certificate authority, and so on), and then the receiving domain controller extracts the *objectGuid* value of the sender's computer object. It then attempts to map that GUID to a server object in its forest. If mapping succeeds, the receiving domain controller additionally verifies that the sending server is an Active Directory domain controller (on the basis of the sending server having a child NTDS Settings object of the proper object class, which can be created only by the directory service itself). Blanket replication access is thereby granted to all Active Directory domain controllers in the forest - and only to Active Directory domain controllers.

For information about certificate security, see "Windows 2000 Certificate Services and Public Key Infrastructure" in this book. For information about Active Directory authentication, see "Authentication" in this book. For information about Active Directory authorization, see "Access Control" in this book.

Advanced Replication Management

Active Directory replication occurs automatically and reliably with no administrative intervention, other than that required to configure sites and site links. Some replication events, however, warrant additional understanding for those administrators who need to fine-tune replication beyond the default behavior. For more information about advanced replication management and troubleshooting, see "Active Directory Diagnostics, Troubleshooting, and Recovery" in this book.

Reciprocal Replication

Replication between sites exhibits request-pull behavior (the receiver, or destination, requests changes from the sender, or source, according to a schedule). Replication within a site, on the other hand, exhibits notify-pull behavior (the receiver is notified of changes by the sender, and the receiver then requests changes from the sender). For replication to occur in two directions, both sides of a connection must be able to initiate replication with the other side.

In cases where replication between sites can be initiated by only one side of a site link, such as when a dial-up connection must go through an Internet service provider (ISP), a flag can be set on the connection object (on the basis of site link attribute information) to implement two-way replication between the source and destination domain controllers of the connection as follows: The domain controller on the dial-up side of the link opens a connection and initiates replication (requests changes). After it receives the changes from the domain controller it contacted, it responds by sending a change notification. The change notification prompts the second domain controller to request changes from the first domain controller. The effect is two-way replication over the initial connection that was opened by the dial-up side of the site link.

The most common scenario in which reciprocal replication is enabled is replication between the main office and a branch office of a company. In this scenario, the branch office must dial up an ISP before a tunneled IP connection (also called a virtual private network or VPN link) can be established. In this example, the following occurs:

1. The branch office dials up the ISP.
2. The branch office establishes a tunneled IP connection to the main office.
3. The branch office initiates replication by requesting changes from the main office.
4. After replication, the branch office immediately sends a change notification to the main office.
5. The main office requests replication changes from the branch office.
6. The branch office replicates its changes to the main office.

In this scenario, reciprocal replication is required because the main office cannot instruct the ISP to first dial up the branch office and, therefore, cannot initiate a connection. Only the branch office can initiate communication to the main office. With reciprocal replication enabled, after the VPN link has been established, replication from the branch office can be initiated by the branch office.

Enabling reciprocal replication between two sites involves modifying the *options* attribute value on the site link object. With this attribute set on the site link, the KCC creates the connections across the link with the appropriate setting that is in effect. Use ADSI Edit to enable reciprocal replication.

To enable reciprocal replication between two sites

1. In ADSI Edit, expand the Configuration container.
2. Navigate to the **Inter-SiteTransports** container, and select **CN=IP**. (You cannot enable reciprocal replication for SMTP links.)
3. Right-click the site link object for the sites for which you want to enable reciprocal replication, 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 **2** in the **Edit Attribute** box.

   If the **Value(s)** box already 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 2. For example, if the value in the **Value(s)** box is 1, calculate 0001 OR 0010 to equal 0011. Type the integer value of the result in the **Edit Attribute** box; for this example, the value is 3.
6. Click **OK**.

Change Notification

Change notification is a mechanism by which a domain controller notifies a replication partner that it has changes. Replication within a site occurs as a response to changes; as changes occur on one domain controller, it notifies its replication partner, which prompts the partner to request the changes. When a domain controller performs an update to an attribute, it sends notification to its replication partner within a specified time following the change.

Change Notification Within a Site

For changes that occur within a site, there is a "holdback timer" that determines the interval between the time a change is made and the time that the source server notifies its replication partners. 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; when the timer expires, the domain controller notifies all of its replication partners (for that directory partition and within the site) that it has changes. If a partner is not engaged in requesting changes from another partner, it sends its change request to the notifying server.

The default value for the holdback timer is 300 seconds, or 5 minutes. To change the default registry setting, you can set a new value in the **Replicator notify pause after modify (secs)** entry in HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \NTDS \Parameters.

Note Very small values for this timer generate redundant notifications, which can decrease performance.

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 30 seconds. To change the default delay, set a new value in the **Replicator notify pause between DSAs (secs)** entry in HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \NTDS \Parameters.

Caution Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. There are programs available in Control Panel or Microsoft Management Console (MMC) for performing most administrative tasks. These programs provide safeguards that prevent you from entering conflicting settings or settings that are

likely to degrade performance or damage your system. Registry editors bypass the standard safeguards that are provided by these administrative tools. Modifying the registry is recommended only when no administrative tool is available. Before you make changes to the registry, it is recommended that you back up any valuable data on the computer. For instructions about how to edit registry entries, see Help for the registry editor that you are using. For more information about the registry, see the *Microsoft Windows 2000 Resource Kit* Technical Reference to the Windows 2000 Registry (Regentry.chm).

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, you can configure change notifications on connections between sites. By modifying the site link object, you can enable change notification 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**. (You cannot enable change notification for SMTP links.)
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.
6. 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.

Urgent Replication

Urgent replication is implemented by immediately notifying replication partners over RPC/IP 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 five minutes. Therefore, if you have change notification enabled on a site link, urgent replication is possible between sites for events that trigger it.

Events That Trigger Urgent Replication

Urgent Active Directory replication is always triggered by certain events on all domain controllers within the same site. When you have enabled change notification between sites, these triggering events also replicate immediately between sites.

Immediate replication between Windows 2000-based domain controllers in the same site is prompted by the following:

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

Urgent Replication of Account Lockout Changes

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

In Windows 2000, account lockout is urgently replicated to the primary domain controller (PDC) emulator role owner and is then urgently replicated to the following:

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

In addition, when authentication fails at a domain controller other than the PDC emulator, the authentication is retried at the PDC emulator. For this reason, the PDC emulator locks the account before the domain controller that handled the failed-password attempt if the bad-password-attempt threshold is reached. For more information about how the PDC emulator role owner manages password changes and account lockouts, see "Managing Flexible Single-Master Operations" in this book.

Managing Urgent Replication

The following guidelines can be useful when deciding whether to enable change notification between sites relative to achieving urgent replication.

- If you want urgent replication everywhere, put all domain controllers for the specific domain in a single site (this option might not be realistic).
- If you want urgent replication everywhere but still want the benefits of site affinity, use multiple sites and enable change notification on all site links.
- By default, a user lockout prompts urgent replication at the site that contains the domain controller that handled the authentication and the site that contains the PDC emulator role owner.

Forced Replication Between Two Servers

You can use a connection object to force replication from the inbound server. In Active Directory Sites and Services, right-click a connection object, and then click **Replicate Now**. For information about how to force replication between two servers by using a connection object, see Windows 2000 Server Help. For more information about forcing replication by using other tools, see "Active Directory Diagnostics, Troubleshooting, and Recovery" in this book.

Replication of Password Changes

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

In Windows 2000 domains, a single domain controller per domain holds the role of PDC emulator, which simulates the behavior of a Microsoft Windows NT version 3.*x*-based or Windows NT 4.0-based primary domain controller. In Windows NT 4.0, the only domain controller that can accept updates is the primary domain controller. If authentication fails at a backup domain controller, the authentication request is passed immediately to the primary domain controller, which is guaranteed to have the current password.

In Windows 2000, when a user password is changed at a specific domain controller, that domain controller attempts to update the respective replica at the domain controller that holds the PDC emulator role. Update of the PDC emulator occurs immediately, without respect to schedules between sites on site links. The updated password is propagated to other domain controllers by normal replication within a site. When the user logs on to a domain and is authenticated by a domain controller that does not have the updated password, the domain controller refers to the PDC emulator to check the credentials of the user name and password rather than denying authentication based on a nonvalid password. Therefore, the user can log on successfully even when the authenticating domain controller has not yet received the updated password.

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

For clients that are running Windows NT 4.0 or clients that are running Microsoft® Windows® 95 or Microsoft® Windows® 98 without the Directory Services Client Update Pack, the client attempts to contact the PDC emulator. If the client is directory-aware, the client contacts any domain controller and the contacted domain controller then attempts to contact the PDC emulator.

Note A domain controller can be set to not contact the PDC emulator if the PDC emulator role owner is not in the current site. If the **AvoidPdcOnWan** entry in HKEY_LOCAL_MACHINE \CurrentControlSet \Services \Netlogon \Parameters is set to 1, the password change reaches the PDC emulator non-urgently through normal replication.

Creation of Extra Connection Objects

The KCC is designed to produce a topology that provides low replication latency, that adapts to failures, and that does not need modification. Adding connections is not recommended because extra connections gradually reduce the ability of the KCC to automatically choose the best configurations. In addition, you create a situation where you must continually evaluate whether the manual connections are doing the best possible job of replicating changes.

Adding extra connections does not necessarily reduce replication latency. Within a site, latency issues are usually related to factors other than the KCC's choices for topology. Factors that affect latency include the following:

- Interruption of the service of key domain controllers, such as the PDC emulator, Global Catalog servers, or bridgehead servers.
- Domain controllers that are too busy replicate in a timely manner (too few domain controllers).
- Network connectivity issues.
- DNS server problems.
- Inordinate amounts of directory updates.

For problems such as these, creating a manual connection does not improve replication latency. Adjusting the scheduling and costs that are assigned to the site link is the best way to influence intersite topology.

The KCC automatically creates connections to keep your directory connected, even if extended failures and outages occur. Create connections manually only if the connections that are automatically configured by the KCC do not connect specific domain controllers that you want to connect. For example, if you want to create a connection on a short-term basis in order to force replication from one particular server, you can use Active Directory Sites and Services to create a connection object if one does not already exist for that server.

Note The cost of the extra connections, as compared to the cost associated with the default configuration, is extra CPU cycles, disk reads, and network messages expended on replication.

You might need extra connections between domain controllers that are within a site or between sites in the following situations:

- When you want to reduce the number of hops between domain controllers within a site. By default, an update takes at most three hops from where it originates to another domain controller in the site. To reduce the number of hops to two hops or to one hop, add extra connections.
- When failures occur between domain controllers in different sites. If failures occur, you might want to add connections that bypass the failed server or servers. (The KCC creates new connections in such cases, but this adaptive process adds some replication latency.)

If you create a connection that is similar to one that the KCC would have created (that is, the KCC would have connected the same two domain controllers in the same direction), the KCC does not create another connection between those servers, nor does it delete or modify the connection that you have created. (For information about how to create a connection object, see Windows 2000 Server Help.)

In large networks, use the following general guidelines to manage connections for optimum KCC performance:

- Reduce the number of sites when possible.
- Increase the memory in your domain controllers.
- Turn off the **Bridge all sites** option, if possible.
- As a last resort, turn off automatic generation of intersite topology and create connections manually.

For more information about managing replication by using extra connections and about KCC scaling recommendations, see the Microsoft Knowledge Base link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources . Search the Knowledge Base by using the keyword "Q244368."
