# How FRS Works

**In this section**



File Replication service (FRS) is a technology that replicates files and folders stored in the SYSVOL shared folder on domain controllers and Distributed File System (DFS) shared folders. When FRS detects that a change has been made to a file or folder within a replicated shared folder, FRS replicates the updated file or folder to other servers. Because FRS is a multimaster replication service, any server that participates in replication can generate changes. In addition, FRS can resolve file and folder conflicts to make data consistent among servers.

These sections provide an in-depth view of how FRS works in an optimal environment. An optimal environment for FRS is defined as follows:

- DNS and Active Directory replication are working properly.
- FRS Active Directory objects are properly configured and present on all domain controllers.
- Kerberos authentication is working properly.
- No firewalls prevent replication from working.
- All servers that participate in replication (known as replica members) have adequate free space on NTFS file system volumes to support the FRS physical structures.
- All replica members have hardware that is sized appropriately for the roles that the servers perform and the hardware is functioning properly.
- No replica members have network, disk, RAM, or CPU bottlenecks.
- All members have the pre-SP1 release of Ntfrs.exe (as described in article 823230, "Issues That Are Fixed in the Pre-Service Pack 1 Release of Ntfrs.exe," in the Microsoft Knowledge Base).
- For SYSVOL replica sets, the number of domain controllers in a domain falls well below the recommended limit of 1,200 domain controllers.
- No replica members are running disk defragmentation, backup, or antivirus software that is known to be incompatible with FRS.
- The update sequence number (USN) journal and staging folder on all replica members are sized according to Microsoft's recommendations.
- The system clocks on all replica members are set to within 30 minutes of each other.
- No files are blocking replication by being held open for an extended period of time on any replica member.
- Changes to a particular file are only made on one replica member so that no conflict resolution is required.
- Administrators do not attempt to copy files from one replica member to another in an attempt to help replication.

---

Back to Top

## FRS Terminology

Before you review the FRS components and processes, it is helpful to understand FRS terminology. The following sections provide a brief introduction and illustration of the basic components of FRS as well as a glossary of terms.

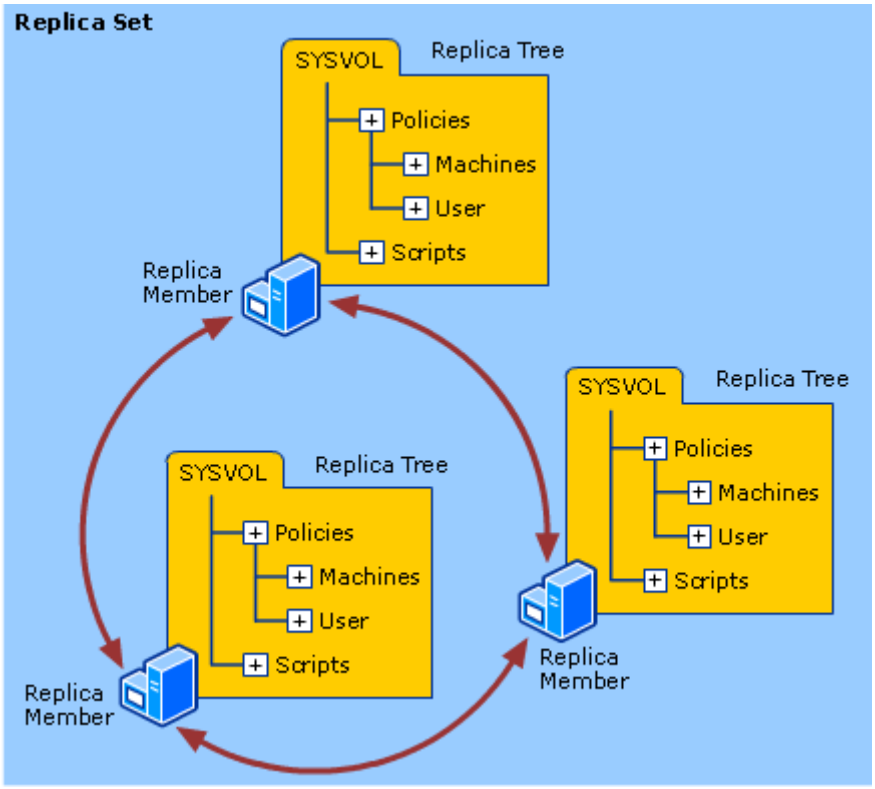
### Basic FRS Components

When you set up DFS replication, you choose a shared folder whose contents you want to replicate. For the SYSVOL shared folder on domain controllers, replication is established automatically during the domain controller promotion process. The set of data to be replicated to all servers is known as the replica tree, and

the servers to receive this replica tree are known as replica members. The group of replica members that participates in the replication of a given replica tree is known as a replica set. Replica members are connected in a topology, which is a framework of connections between servers and defines the replication path. There are a number of common topologies, such as full mesh, ring, and hub and spoke.

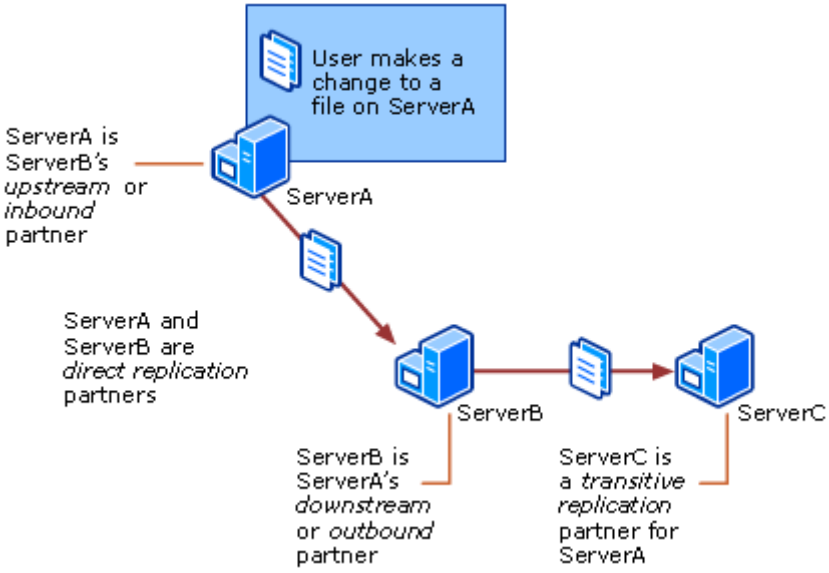
The following figure illustrates these terms.

**Basic FRS Components (Ring Topology)**




It is also helpful to understand the terminology used to describe the relationship between two members. This terminology is illustrated in the following figure.

**Replication Partner Terminology**




As illustrated in the figure, any two replica members with a direct connection between them are known as direct replication partners. In the figure, ServerA and ServerB are direct replication partners, as are ServerB and ServerC. Although changes that originate on ServerA are eventually replicated to ServerC, these two servers do not have a direct connection to each other, so they are known as transitive replication partners.

When a changed file or folder is received on a server, either because the change originated there or because the server received the file or folder from another partner, that server becomes the upstream partner, also known as inbound partner, for any direct replication partner that has not yet received the change. For example, in the figure titled "Replication Partner Terminology," a change originates on ServerA, and ServerA is considered the upstream or inbound partner for ServerB. ServerB is considered the downstream (also known

as outbound) partner for ServerA.

### Glossary of FRS Terms

The following terms are used to describe the components and processes of FRS.

**Active Directory object** A distinct set of named attributes that represents a network resource. FRS uses Active Directory objects to represent servers that participate in replica sets and the topology that FRS uses to replicate data.

**Authoritative restore (also called D4)** The process whereby one replica member is made the authoritative member, and all the data in its replica tree is replicated to all other replica members. This procedure should be used only in extreme circumstances under the supervision of your support provider or Microsoft Product Support Services. This process is informally referred to as a "D4" in reference to the registry setting used to invoke this process.

**Burflags** Short for "backup restore flags," an FRS-related registry entry used to change the behavior of the FRS service at startup. The values for this registry entry are D2 (for nonauthoritative restores) and D4 (for authoritative restores).

**Change order** A message that contains information about a file or folder that has changed on a replica member. The change order is sent to the member's outbound partners. If the outbound partners accept the change, the partners request the associated staging file. After installing the changed file in their individual replica trees, the partners propagate the change order to their outbound partners.

**Directed change order** A change order that is directed to a single outbound partner and primarily produced when the partner is doing a vvjoin, such as during initial synchronization or a nonauthoritative restore (D2).

**File event time** The time at which a change to a file or folder is made. This might not be the same as the file create or last-write time. For example, restoring a file from a backup tape preserves the file create and last-write times, but the file event time is the time when the actual file restoration was performed.

**File GUID** An identifying property of a file or folder in a replica tree. FRS creates and manages file globally unique identifiers (GUIDs), which, along with the replication version number and event time, are stored in the file ID table in the FRS database. Each file and folder stores its file GUID as part of its NTFS attributes; therefore, corresponding files and folders across all replica set members have the same file GUID.

**File ID table** A table in the FRS database that contains an entry with version and identity information for each file and folder in the replica tree.

**File version number** A property of a file and folder in a replica tree that is incremented each time the file or folder is updated. The file version number is used to resolve concurrent updates originating from more than one member of the replica set. The version number is only incremented by the member that originated the file update. Other members that propagate the update do not change the version number.

**Filter** A setting that excludes subfolders (and their contents) or files from replication.

**FRS debug logs** Text files in the *systemroot*\Debug folder that store FRS transaction and event detail.

**Inbound connection** For a given replica member, a component of the NTFRS Member object in Active Directory that identifies inbound partners. An inbound connection exists for each inbound partner.

**Inbound log** A table in the FRS database that stores pending change orders to be processed. As entries are processed, acknowledgments are sent to the inbound partners.

**Knowledge Consistency Checker (KCC)** A built-in process that runs on all domain controllers and generates replication topology for the Active Directory forest. The KCC creates separate replication topologies depending on whether replication is occurring within a site (intrasite) or between sites (intersite). The KCC also dynamically adjusts the topology to accommodate new domain controllers, domain controllers moved to and from sites, changing costs and schedules, and domain controllers that are temporarily unavailable.

**Link target** In a DFS namespace, the mapping destination of a link, typically a shared folder on a server. FRS can be used to keep link targets (shared folders) synchronized on different servers.

**Local change order** A change order that is created because of a change to a file or folder on the local server. The local server becomes the originator of the change order and constructs a staging file.

**MD5 checksum** A cryptographically secure one-way hashing algorithm that is used by FRS to verify that a file on each replica member is identical.

**Morphed folder** A folder that FRS creates when resolving a name conflict between two or more identically named directories that originate on different replica members. FRS identifies the conflict during replication, and the receiving member protects the original copy of the folder and renames (morphs) the later inbound copy of the folder. The morphed folder names have a suffix of "_NTFRS_xxxxxxxx," where "xxxxxxxx" represents eight random hexadecimal digits.

**Nonauthoritative restore (also called D2)** The process whereby a given replica member is reinitialized by obtaining a replica tree from another replica member. This process is often used to resynchronize a replica member's replica tree with one of its inbound partners, such as after a server failure. This process is informally referred to as a "D2" in reference to the registry setting used to invoke this process.

**Originator GUID** A globally unique identifier (GUID) that is associated with each replica member. All change orders produced by a given replica member carry the replica member's originator GUID, which is saved in the file ID table.

**Outbound connection** For a given replica member, a component of the NTFRS Member object in Active Directory that identifies outbound partners. An outbound connection exists for each outbound partner.

**Outbound log** A table in the FRS database that stores pending change orders to be sent to outbound partners. The changes can originate locally or come from an inbound partner. These change orders are eventually sent to all outbound replica partners.

**Parent GUID** The globally unique identifier (GUID) of the parent folder that contains a particular file or folder in the replica tree.

**Preinstall folder** A hidden subfolder under the root of the replica tree. When a newly created file or folder is replicated to a downstream partner, the file or folder is first created in the preinstall folder. After the file or folder is completely replicated, it is renamed to its target location in the replica tree. This process is used so that partially constructed files are not visible in the replica tree.

**Reanimation change order** A change order that FRS uses to reconcile a recently received change order against a previously deleted file or folder.

**Remote change order** A change order received from an inbound (or upstream) partner that originated elsewhere in the replica set.

**Retry change order** A change order that is in some state of completion but was blocked for some reason and must be retried later.

**Schedule** The frequency at which data replicates.

**Staging folder** A folder that acts as a queue for changed files and folders to be replicated to downstream partners.

**Staging file** A file that FRS constructs in the staging folder as a result of a file or folder change. FRS constructs staging files by using backup application programming interfaces (APIs) and then replicates the staging files to downstream partners. Downstream partners use restore APIs to reconstruct the staging files in the preinstall folder before renaming the files into the replica tree. Staging files are compressed to reduce the network bandwidth used during replication.

**SYSVOL** On a domain controller, a shared folder that stores a copy of the domain's public files, including system policies and Group Policy settings, that are replicated to all other domain controllers in the domain by FRS.

**Update sequence number (USN)** A monotonically increasing sequence number for each volume. The USN is incremented every time a modification is made to a file on the volume.

**USN journal** On NTFS volumes, a persistent log that tracks all changes on the volume, including file creations, deletions, and changes. The USN journal has a configurable maximum size and is persistent across reboots and system crashes. FRS uses the USN journal to monitor changes made in the replica tree.

**USN journal wrap** An error that occurs when large numbers of files change so quickly that the USN journal must discard the oldest changes (before FRS has a chance to detect the changes) to stay within the specified size limit. To recover from a journal wrap, you must perform a nonauthoritative restore on the server to synchronize its files with the files on the other replica members.

**Version vector** A vector of volume sequence numbers (VSNs), with one entry per replica set member. All change orders carry the originator GUID of the originating member and the associated VSN. As each replica member receives the update, it tracks the VSN in a vector slot that is assigned to the originating member. This vector describes whether the replica tree is current with each member. The version vector is then used to filter updates from inbound partners that might have already been applied. The version vector is also delivered to the inbound partner when the two members join. When a new connection is created, the version vector is used to scan the file ID table for more recent updates that are not seen by the new outbound partner.

**Version vector join (vvjoin)** The process in which a downstream partner joins with an upstream partner for the first time.

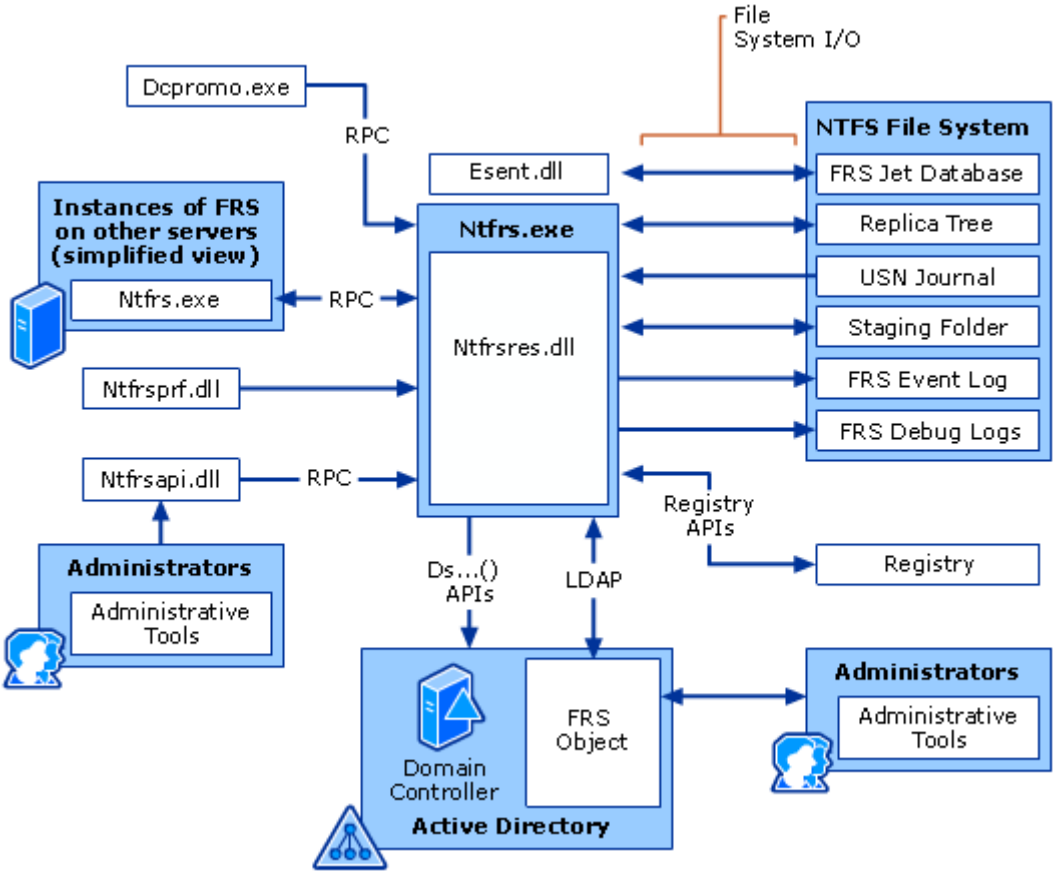
---

Back to Top

## FRS Architecture

The File Replication service (Ntfrs.exe) is the primary component of the FRS architecture. This service and its related components are illustrated in the following figure.

**FRS Architecture**




The following table describes the components related to the FRS architecture.

**Components of the FRS Architecture**

| Component | Description |
|---|---|
| Active Directory | Where FRS topology and schedule information is stored. FRS periodically polls Active Directory to retrieve updated information about the FRS topology and schedule. FRS uses Lightweight Directory Access Protocol (LDAP) to connect to Active Directory using port 389. |
| Esent.dll | The dynamic link library of the Jet database. |
| File Replication service event log (NtFrs.Evt) | The log where events and error messages related to FRS are logged in Event Viewer. The FRS event log file is located in *Systemroot*\System32\Config\. |
| FRS debug logs | Text files where FRS stores transaction and event details. The FRS debug logs are typically used for troubleshooting FRS. |
| FRS Jet database (Ntfrs.jdb) | The database where FRS transactions are stored. |
| Ntfrs.exe | The executable file for the File Replication service. Runs as LocalSystem on all domain controllers and members of DFS replica sets. |
| Ntfrsapi.dll | The dynamic link library file that allows other applications to get information about FRS via remote procedure call (RPC). These applications include Active Directory Installation Wizard (Dcpromo), Ntfrsutl.exe (part of the Windows Support Tools), Windows Backup, and Volume Shadow Copy service. |
| Ntfrsprf.dll | The dynamic link library file that allows System Monitor to connect to FRS and collect performance-related information via local remote procedure call (LRPC). |
| Ntfrsres.dll | The dynamic link library file where event log messages are stored. When events occur, the event log messages are displayed in the File Replication service event log. |
| Registry | Where FRS configuration data is stored. Some FRS-related registry keys are recognized when changes to those keys occur; other registry keys are read only when the File Replication service is first started or restarted. |
| Replica tree | The contents of a folder that is replicated among replica members in a replica set. |
| Staging folder | A folder that acts as a queue for changed files to be replicated to downstream partners. |
| USN journal (also | A persistent log of changes made to files on an NTFS volume. NTFS uses the change |

| | |
|---|---|
| called NTFS change journal) | journal to track information about added, deleted, and modified files for each volume. The USN journal describes the nature of any changes to files on the volume. When any file or folder is created, modified, or deleted, NTFS adds a record to the USN journal for that volume. |

---

Back to Top

## FRS Protocols

FRS stays at application layer of Open Systems Interconnection (OSI) model, and uses RPC as transport medium for communication between replica members.

---

Back to Top

## FRS Interfaces

Windows Server 2003 includes an FRS writer for the Volume Shadow Copy service. The FRS writer allows Volume Shadow Copy service-compatible backup programs, such as Windows Backup, to make point-in-time, consistent backups of the replica tree. Using a Volume Shadow Copy service-compatible backup program ensures that replica tree backups do not contain partial files (which can occur when a replica member receives an updated file and begins overwriting the previous version of the file in the replica tree with the updated version of the file). This process, known as the install process, creates a window during which the file is half-written. The window exists because FRS must move the updated file from the preinstall folder and overwrite the existing file in the replica tree.

Before a Volume Shadow Copy service-compatible backup program takes a shadow copy of a replica tree, the program instructs FRS to stop requesting new work items from the install command server queue. After all currently active installations are complete, FRS enters a frozen state during which no installations occur. Shadow copies made while FRS is in a frozen state will not contain partial files. The install service thaws either when the backup program signals a thaw or at a predetermined time-out period.

For more information about the Volume Shadow Copy service, see "Volume Shadow Copy Service Technical Reference."

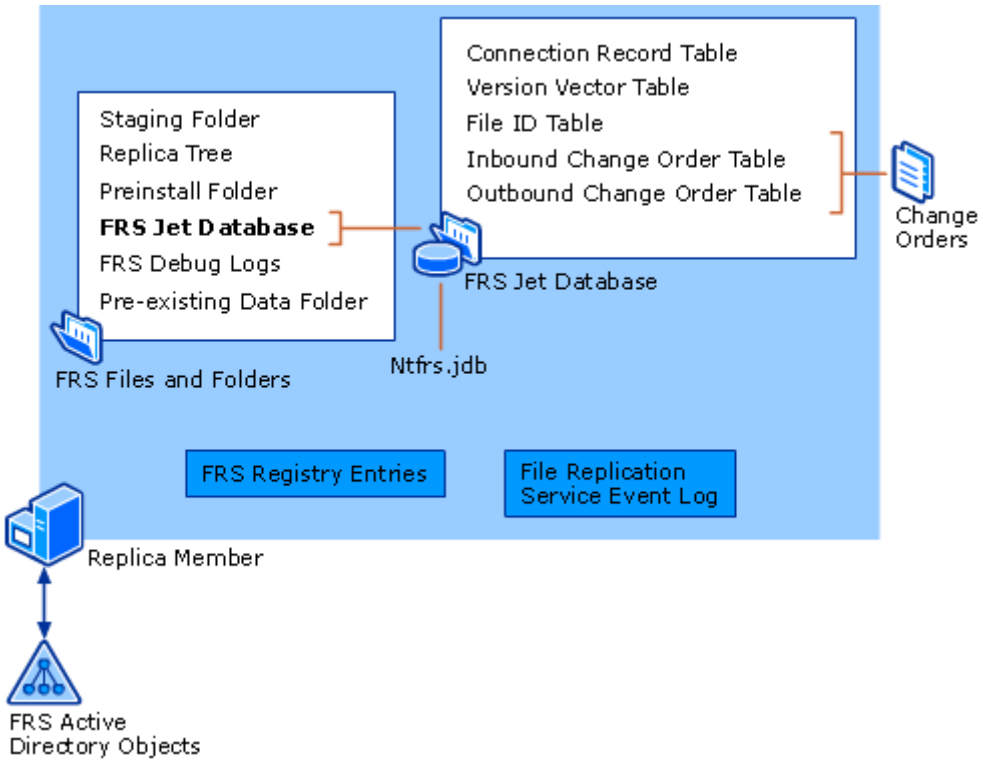
---

Back to Top

## FRS Physical Structures

FRS requires a number of physical structures to be stored in Active Directory and on each replica member. These structures are illustrated in the following figure and are described in the sections that follow.

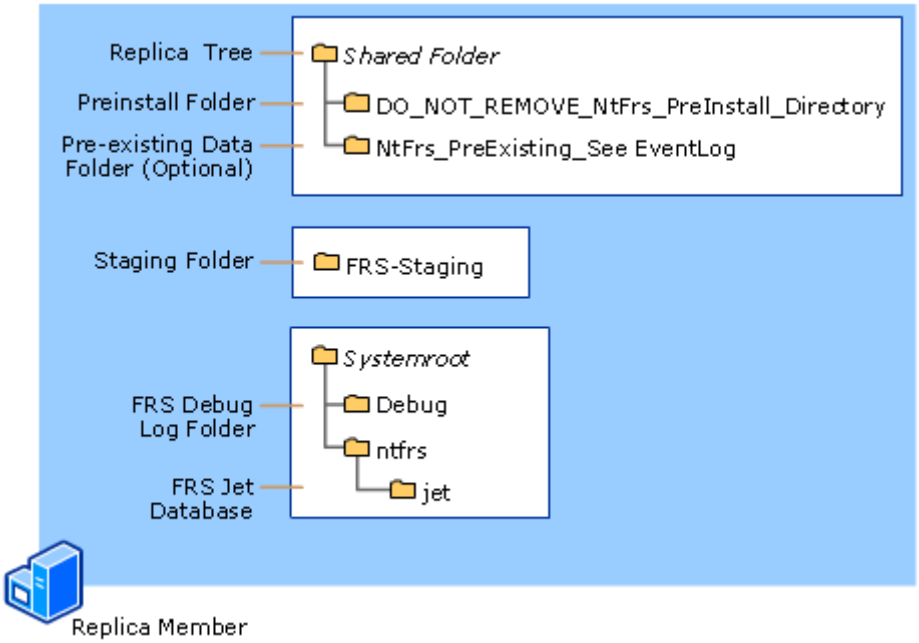
**Overview of FRS Physical Structures**




**FRS Files and Folders on Replica Members**

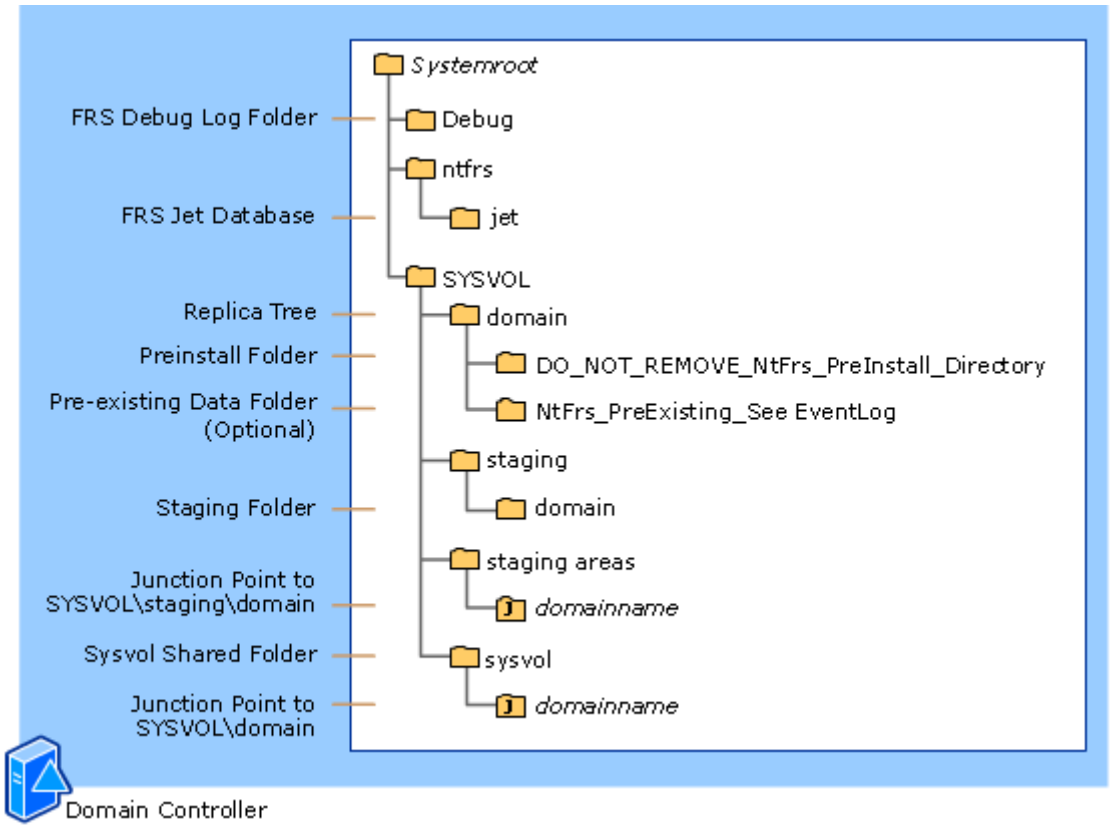
Each replica member contains a number of files and folders used by FRS for SYSVOL and DFS replica sets. The following figure illustrates the files and folders that are stored on a DFS replica member.

**FRS Files and Folders on DFS Replica Members**




The following figure illustrates the files and folders stored on domain controllers. Note how the location of these folders differs from DFS replica members.

**FRS Files and Folders on Domain Controllers**




The following sections briefly describe each of these folders.

### Replica tree on DFS replica members

The replica tree is the set of files and folders that are kept in sync on all replica members. The replica tree begins at the replica root, which is the shared folder where all the replicated data is stored. The replica tree can exist on any NTFS volume for DFS replica members, and FRS does not impose limits on the size of the replica tree.

### SYSVOL folder domain controllers

SYSVOL is a folder that stores numerous FRS-related folders and a copy of the domain's public files, including system policies and Group Policy settings that are replicated to all other domain controllers in the domain by FRS. The actual replica root begins at the *systemroot*\SYSVOL\domain folder, but the folder that is actually shared is the *systemroot*\SYSVOL\sysvol folder. These folders appear to contain the same content because SYSVOL uses junction points (also called reparse points). A junction point is a physical location on a hard disk that points to data that is located elsewhere on the hard disk or on another storage device. The following table lists the folders in SYSVOL that contain junction points and the locations to which these junction points resolve:

**SYSVOL Junction Points**

| Location of Junction Point | Where the Junction Point Resolves |
|---|---|
| SYSVOL\staging areas*domainname* | SYSVOL\staging\domain |
| sysvol*domainname* | SYSVOL\domain |

### Staging folder

The staging folder is a hidden folder that acts as a queue for changed files to be replicated to downstream partners. After the changes are made to a file and the file is closed, FRS creates the file in the staging folder by using backup application programming interfaces (APIs) and replicates the file according to schedule. Files in the staging folder are called staging files. Any further use of corresponding files in the replica tree does not prevent FRS from replicating the staging files to other members. In addition, if the file is replicated to multiple downstream partners or to members with slow data links, using staging files ensures that the underlying files in the replica tree can still be accessed.

The default size of the staging folder is approximately 660 megabytes (MB), the minimum size is 10 MB, and the maximum size is 2 terabytes. For performance and disk space purposes, the staging folder can be located on a volume other than where the replica tree is stored.

For more information about the staging folder, see "How the Staging Folder Works" later in this section.

### Preinstall folder

The preinstall folder is a hidden system folder named DO_NOT_REMOVE_NtFrs_PreInstall_Directory. When a downstream partner begins receiving a new file (or folder), the file is installed in the preinstall folder on the downstream partner so that a partially constructed file is not added to the replica tree. After the downstream partner receives the entire file in the preinstall folder, the file is renamed to its target location in the replica tree.

### FRS Jet database

FRS creates a Microsoft Jet database at *systemroot*\ntfrs\jet\ to store FRS transactions. FRS does not impose size limits on the Jet database, but replication stops working if the volume where the database is stored runs out of disk space. For performance and disk space purposes, the Jet database can be moved to a volume other than where the replica tree is stored.

For more information about the FRS Jet database, see "FRS Database Tables" later in this section.

### FRS debug logs

FRS creates debug logs in the *systemroot*\Debug folder. By default, the debug logs store FRS transaction and event detail in sequentially numbered files from Ntfrs_0001.log through Ntfrs_0005.log. The characteristics of the debug logs are determined by the values of several registry entries. These values allow you to set the number of log files used, the number of entries per log, the level of detail logged, and so forth.

FRS creates another log file, Ntfrsapi.log, to track events that occur during the promotion and demotion of domain controllers. Information in the Ntfrsapi.log file includes which server was chosen as the parent for the initial replication (seeding) process and the creation of registry keys. Because errors during the promotion and demotion processes are also tracked in this file, Ntfrsapi.log is useful for troubleshooting problems where the SYSVOL and Netlogon shared folders are not shared correctly.

For performance and disk space purposes, the debug logs can be moved to a volume other than the one on which the replica tree is stored.

For more information about the FRS debug logs, see "How FRS Debug Logs Work" later in this section.

### Pre-existing data folder

The pre-existing folder, named NtFrs_PreExisting___See EventLog, is an optional folder that is located under the replica root. If the pre-existing data folder is present on a replica member, FRS created it after one of the following events:

- The server was added to a replica set but the server already had one or more files in the shared folder that became the replica tree. In this case, FRS moved that data into the pre-existing data folder and then replicated the replica tree from one of the upstream partners to the new member.
- The replica member had a nonauthoritative restore (also called D2) performed on it by an administrator. This process is used to bring a replica member back up to date with its partners after problems such as assertions in the FRS service, corruption of the local FRS Jet database, journal wrap errors, and other replication failures. When you perform a nonauthoritative restore on a server, FRS moves the existing data in the replica tree to the pre-existing data folder and then receives the replica tree from one of the upstream partners.
- The server was prestaged before it was added to the replica set. During the prestaging process, files in the replica tree are temporarily moved from the replica tree to the pre-existing data folder. For more information about the prestaging process, see "How Prestaging Works" later in this section.

Only one pre-existing data folder can exist at a time. If one of the previously listed events occurs, causing the pre-existing data folder to be created, and then another one of the events occurs, the previous pre-existing data folder is deleted and replaced with another pre-existing data folder.

### FRS Database Tables

The Jet database (Ntfrs.jdb) used by FRS has a set of five tables:

- Connection record table
- Version vector table
- File ID table
- Inbound change order table
- Outbound change order table

You can view the contents of three of these tables (the file ID table, inbound change order table, and outbound change order table) by using Ntfrsutl.exe (a Windows Support Tool), though doing so is typically only necessary for troubleshooting purposes. Windows Server 2003 provides several Windows Support Tools scripts, Topchk.cmd, Connstat.cmd, and Iologsum.cmd, that can present the output of Ntfrsutl in a more readable format.

The following sections briefly describe the tables in Ntfrs.jdb.

### Connection record table

The connection record table tracks the state of each inbound and outbound connection for the replica set and tracks the delivery state of each outbound change order.

### Version vector table

The version vector table contains a vector of volume sequence numbers (VSNs), one entry per replica member. All change orders carry the originator globally unique identifier (GUID) of the originating member and the associated volume sequence number. As each replica set member receives the update, it tracks the VSN in a vector slot assigned to the originating member. This vector now describes how up-to-date this member's replica tree is with respect to each member. The version vector is then used to filter out updates from inbound partners that might have already been applied. The version vector is also delivered to the inbound partner when two members join. When a new connection is created, the version vector is used to scan the file ID table for more recent updates not seen by the new outbound partner.

### File ID table

The file ID table lists all files in the replica sets on the local server of which FRS is aware. Data stored in the file ID table includes the file name, file GUID, NTFS file ID, parent GUID, parent file ID, version number, event time, and originator GUID. You can view entries in the file ID table by using Ntfrsutl.exe with the **idtable** parameter.

### Inbound change order table (inbound log)

The inbound change order table, referred to as the inbound log, stores pending change orders to be processed. As entries are processed, acknowledgments are sent to the inbound partners. State variables in the record track the progress of a change order and determine where to continue after a system crash or a retry of a failed operation. Data stored in the inbound log includes the change order's GUID, file name, originator GUID, file GUID, version number, and event time. You can view change orders in the inbound log by using Ntfrsutl.exe with the **inlog** parameter.

### Outbound change order table (outbound log)

The outbound change order table, referred to as the outbound log, stores pending change orders to be sent to outbound partners. By default, change orders remain in the outbound log for seven days, even if all replication partners have received the change. Also in the outbound log is the leading (next change) and trailing (last acknowledged) index for each partner. You can view change orders in the inbound log by using Ntfrsutl.exe with the **outlog** parameter.

### FRS Change Orders

A change order is a message that contains information about a file or folder that has changed on a replica member. Change orders are stored in the inbound and outbound logs on each replica member and contain information such as the change order's GUID, the originator GUID, the file GUID, the file name, event time, and file version number.

You can view change orders in the inbound and outbound logs by using Ntfrsutl.exe with the **inlog** and **outlog** parameters. You can also use the Windows Support Tools script Iologsum.cmd to parse the Ntfrsutl output into a more readable format.

FRS uses the following five types of change orders:

### Local change order

A change order that is created because of a change to a file or folder on the local server. The local server becomes the originator of the change order and constructs a staging file.

### Remote change order

A change order received from an upstream partner that originated elsewhere in the replica set.

### Retry change order

A change order that is in some state of completion but was blocked for some reason and must be retried later. This is a change order property in that both local and remote change orders can become retry change orders.

### Directed change order

A change order that is directed to a single outbound partner and primarily produced when the partner is performing an initial sync (vvjoin) or a nonauthoritative restore (D2).

### Reanimation change order

A change order that FRS uses to reconcile a recently received change order against a previously deleted file or folder.

### FRS Registry Entries

FRS registry entries are located in the registry under HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NtFrs. For a list of FRS registry entries, see "FRS Tools and Settings."

### File Replication Service Event Log

The File Replication service event log contains events logged by FRS. It is created when FRS is first enabled on a server, and the creation process involves setting registry keys under HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\File Replication Service. The event log file name is NtFrs.Evt, which is located under *systemroot*\System32\Config.

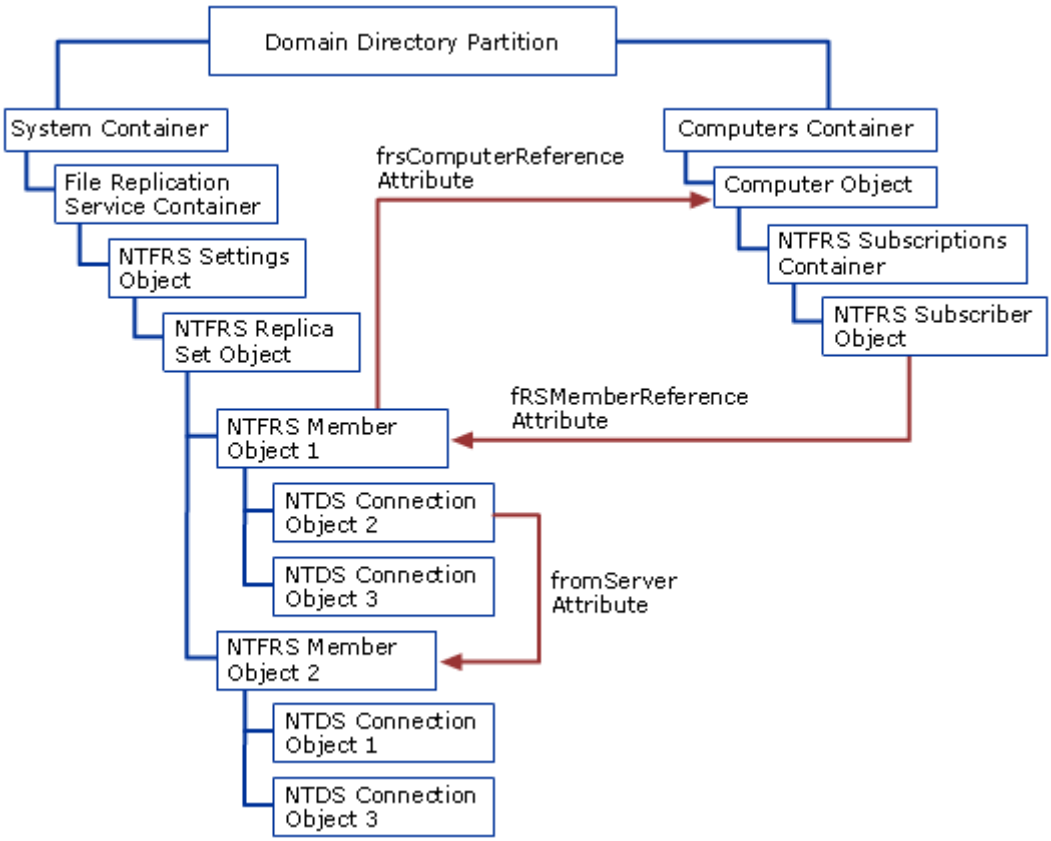
### FRS Objects in Active Directory

For FRS to function properly, certain critical Active Directory objects (as well as their attributes and parent containers) must exist in Active Directory. These objects, which define a replica set's topology, schedule, and filters, are created in Active Directory when you use the Distributed File System snap-in to configure replication for DFS or when you promote a server to a domain controller using the Active Directory Installation Wizard (Dcpromo.exe). Active Directory replication is used to replicate these objects to all domain controllers in a domain, and missing or corrupted objects can cause FRS replication to fail.

### Hierarchy of FRS Objects in Active Directory for DFS Replica Sets

The following figure illustrates the hierarchy of FRS-related Active Directory containers and objects for DFS replica sets. The figure also illustrates how several of these objects are linked together using reference attributes. For example, in the NTFRS Member object, there is an attribute called **frsComputerReference** that refers to the computer object for the server.

**FRS Containers and Objects in Active Directory for DFS Replica Sets**




The following sections briefly describe the Active Directory objects, containers, and their attributes for DFS replica sets. The class name of each object is shown in parentheses.

### NTFRS Settings object

The NTFRS Settings object (class **nTFRSSettings**) is used as a container for the NTFRS Replica Set objects. The NTFRS Settings object can contain other NTFRS Settings objects; therefore, it provides a way to form a hierarchy to better organize the NTFRS Replica Set objects.

### NTFRS Replica Set object

Every NTFRS Replica Set object (class **nTFRSReplicaSet**) represents a set of computers that replicate a specified folder tree and a common set of data among them. There is one NTFRS Replica Set object for every replica set. The NTFRS Replica Set object must be directly under an NTFRS Settings object.

The most commonly used attributes on this object are **fRSReplicaSetType** (set to 3 for DFS replica sets), **fRSFileFilter**, **fRSDirectoryFilter**, and **Schedule**. If you set the **Schedule** attribute, it applies to all the NTDS Connection objects in the replica set that do not have a **Schedule** attribute.

### NTFRS Member object

Every NTFRS Member object (class **nTFRSMember**) corresponds to a computer that is part of the replica set. The relationship between the NTFRS Member object and the computer object is indicated by the **frsComputerReference** attribute. The NTFRS Member object can contain one or more NTDS Connection objects that define the inbound partners that a member replicates from. NTDS Connection objects refer to other member objects in the same replica set object using the **fromServer** attribute.

### NTDS Connection object

NTDS Connection objects (class **nTDSConnection**) define the inbound and the outbound partners of a replica member. NTDS Connection objects are located under the NTFRS Member object in the domain naming context for DFS replica sets.

The NTDS Connection object is inbound to the NTFRS Member object that it is located under, and it is outbound from the NTFRS Member object that its **fromServer** attribute points to.

### Computer object

Each computer object represents a computer in the domain. The relationship between the NTFRS Member object and the computer object is indicated by the **frsComputerReference** attribute.

### NTFRS Subscriptions container

The NTFRS Subscriptions object (class **nTFRSSubscriptions**) is similar to the NTFRS Settings object in that it is primarily used as a container to group NTFRS Subscriber objects. The **fRSWorkingPath** attribute defines the location of the Ntfrs.jdb file, which is typically located in the %SystemRoot%\Ntfrs folder.

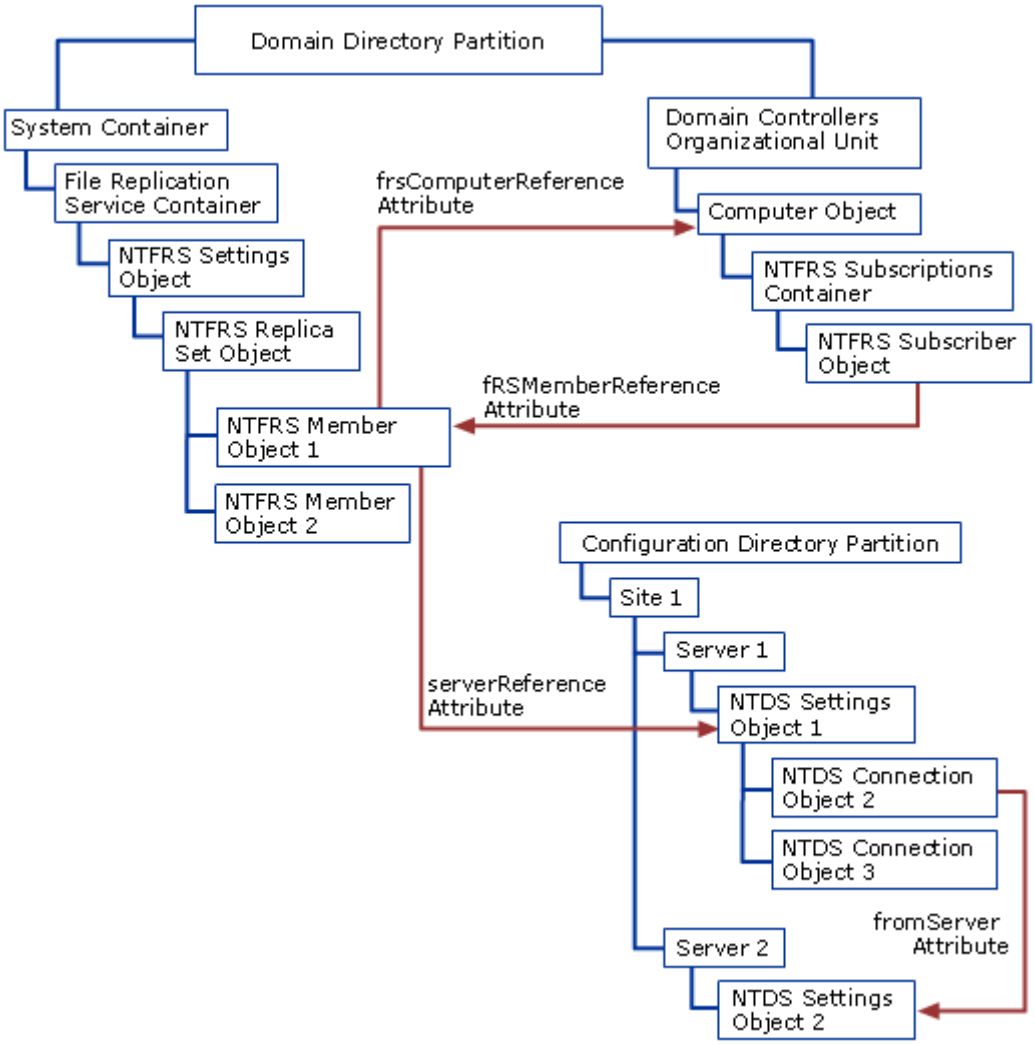
### NTFRS Subscriber object

Every NTFRS Subscriber object (class **nTFRSSubscriber**) under a computer's computer object corresponds to a replica set that the computer is a member of. The **fRSMemberReference** attribute of the NTFRS Subscriber object points to the NTFRS Member object of the replica set that it corresponds to. Every NTFRS Subscriber object also has both an **fRSRootPath** attribute that specifies the folder tree to replicate and an **fRSStagingPath** attribute that specifies the folder to store the staging files under.

### Hierarchy of FRS Objects in Active Directory for SYSVOL Replica Sets

SYSVOL replica sets have a different hierarchy of objects in Active Directory than do DFS replica sets. This hierarchy is illustrated in the following figure.

**FRS Containers and Objects in Active Directory for SYSVOL Replica Sets**




The following sections describe any attributes and relationships that are specific to SYSVOL replica sets. In some cases, the objects for DFS and SYSVOL replica sets are identical.

### NTFRS Settings object

Same as for DFS replica sets.

### NTFRS Replica Set object

Same as for DFS replica sets except that only one NTFRS Replica Set object can be of the SYSVOL type in a domain. In this case, the **fRSReplicaSetType** attribute is set to 2.

### NTFRS Member object

In SYSVOL replica sets, the **serverReference** attribute of the NTFRS Member object points to the NTDS Settings objects (class **nTDSDSA**) that contain the NTDS Connection objects that this member replicates from.

### NTDS Connection object

For SYSVOL replica sets, FRS uses both manually generated connection objects and connection objects that are generated by Knowledge Consistency Checker (KCC) that are located in the NTDS Settings object in the configuration naming context. (You can use the Active Directory Sites and Services snap-in to view these connection objects.) These connection objects are also used during Active Directory replication.

For SYSVOL replica sets, the NTDS Connection object is inbound to the NTFRS Member object that corresponds to the NTDS Settings object that the NTDS Connection object is located under. It is outbound from the NTFRS Member object that corresponds to the NTDS Settings object that its **fromServer** attribute points to.

### Computer object

Same as for DFS replica sets.

### NTFRS Subscriptions container

Same as for DFS replica sets.

**NTFRS Subscriber object**

Same as for DFS replica sets.

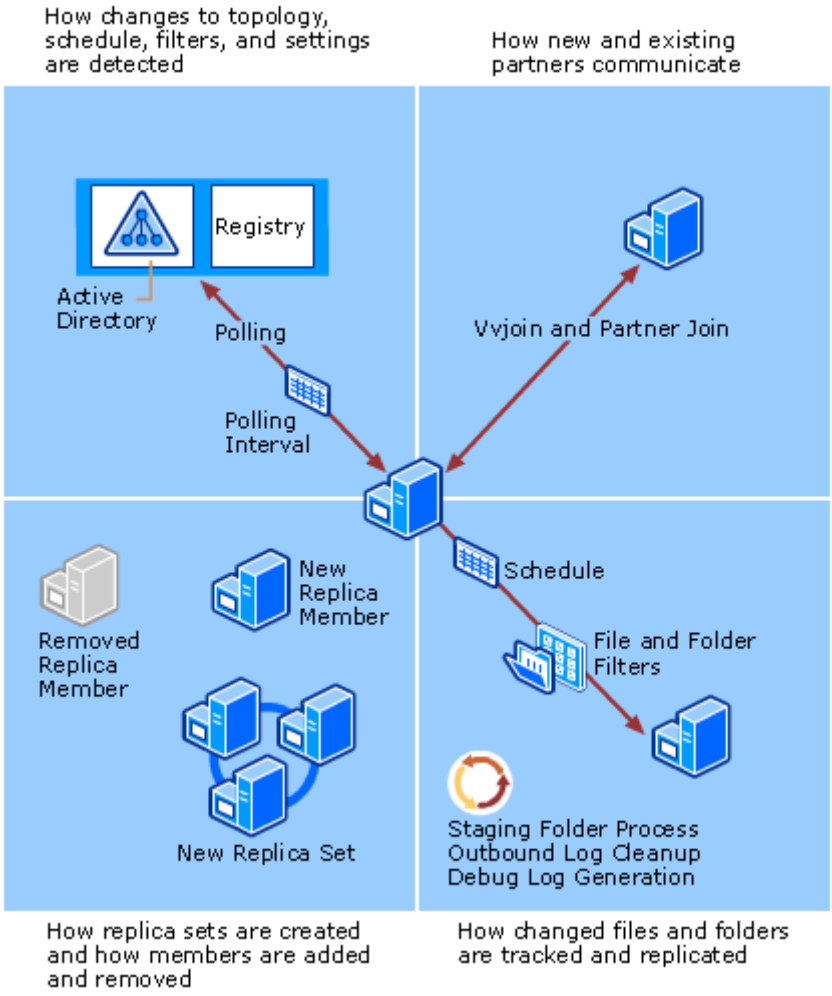
---

Back to Top

## FRS Processes and Interactions

The following figure illustrates four categories of FRS processes and interactions described in this section.

**FRS Processes and Interactions**




The following sections describe the four categories of processes and interactions.

### How changes to topology, schedule, filters, and settings are detected

FRS periodically polls Active Directory to obtain new or updated information about replica set topology, schedule, and file and folder filters. FRS also polls the registry to detect updated settings. For information about the Active Directory and registry polling process and the intervals at which polling occurs, see "How FRS Polling Works" later in this section.

### How new and existing partners communicate

Replica members communicate with each other to determine whether new or changed files and folders need to be replicated from one member to another. When a downstream partner joins an upstream partner for the first time, the process is known as a version vector join (vvjoin). For more information, see "How the Vvjoin Process Works" later in this section. Routine communication between members is described in "How the Partner Join Process Works" later in this section.

### How replica sets are created and how members are added and removed

The creation of new replica sets and the addition and removal of replica members involves many processes. For example, when a new replica member is added to a replica set, the FRS physical structures are created on the member and in Active Directory, the FRS service is started, registry keys are created, the member polls the registry and Active Directory, the new member vvjoins with an upstream partner, and so on.

The following sections describe these processes and interactions:

- "What Happens When You Create a DFS Replica Set"
- "What Happens When SYSVOL Is Created During Domain Controller Promotion"
- "How Prestaging Works"
- "What Happens When You Remove a Member from a Replica Set"

### How changed files and folders are tracked and replicated

A number of processes occur when a file or folder is changed, closed, and then replicated from one member to another. Change orders are generated and flow from one member to another, staging files are built in the staging folder, and file and folder filters prevent files and folders with specified names or extensions from replicating. Replication takes place according to a schedule, change orders are periodically removed from the outbound log, and details about these processes are written to the FRS debug logs. You can find information about these processes in the following sections:

- "Types of Changes That Trigger Replication"
- "How Files and Folders Are Replicated"
- "How the Staging Folder Works"
- "How Replication Schedules Work"
- "How File and Folder Filters Work"
- "How the Outbound Log Cleanup Process Works"
- "How FRS Debug Logs Work"

### How FRS Polling Works

FRS periodically polls Active Directory (using LDAP) to obtain new or updated information about replica set topology, schedules, and file and folder filters. (This information is stored in the Active Directory objects described in "FRS Objects in Active Directory" earlier in this section.) FRS also polls the registry to detect updated settings, such as staging directory size, USN journal size, debug log settings, and so forth. Polling takes place after FRS starts up, such as when the host computer or the service starts (when **net start ntfrs** is typed at the command line, for example) and during regular intervals after that.

The following sections describe the polling process and the intervals at which polling occurs.

### How Active Directory Polling Works

The Active Directory polling process is as follows. Note that if any of these steps fails, replication does not occur. Failure is typically caused by network connectivity issues or missing, damaged, or duplicate Active Directory objects.

1. FRS locates the computer object for the server it is running on and enumerates the NTFRS Subscriber objects.
2. For a given subscriber, FRS follows the **fRSMemberReference** attribute to the NTFRS Member object
3. Next, FRS reads the NTFRS Replica Set object (always directly above the NTFRS Member object) to determine whether it is SYSVOL or DFS replica set.

After FRS determines the replica type (SYSVOL or DFS), FRS enumerates the connection objects. This process is different for SYSVOL and DFS.

### How FRS enumerates connection objects for DFS replica sets

1. FRS finds all the NTDS Connection objects that represent an inbound (or upstream) connection to this member. (All NTDS Connection objects that represent an inbound connection are directly under the NTFRS Member object.)
2. FRS finds all the NTDS Connection objects that represent an outbound (or downstream) connection to this member. (All NTDS Connection objects that represent an outbound connection have this member's name in the **fromServer** attribute.)
3. FRS forms a list of NTFRS Member objects that have an inbound or outbound connection with this member. This list is a combined list for all the replica sets that this server is a member of.
4. FRS forms a list of the computers corresponding to the members in the member list. All members have an **frsComputerReference** attribute pointing to the computer object.
5. FRS finds the DNS name of each of the computers in the computer list. This is the name that FRS uses to bind to its inbound and outbound partners.

### How FRS enumerates connection objects for SYSVOL replica sets

1. FRS finds all the NTDS Connection objects that represent an inbound connection to this member. The **serverReference** attribute on the NTFRS Member object points to the NTDS Settings object. (All the NTDS Connection objects that represent an inbound connection are directly under the NTDS Settings object.)
2. FRS finds all NTDS Connection objects that represent an outbound connection to this member. (All NTDS Connection objects that represent an outbound connection have this member's name in the **fromServer** attribute.)
3. FRS forms a list of member objects that have an inbound connection or outbound connection with this member. Forming the list of members is a two-step process for SYSVOL replica sets. First, FRS forms the list of NTDS Settings objects, and then FRS finds the NTFRS Member objects that point to each of these NTDS Settings objects. This list is a combined list for all the replica sets that this server is a member of.
4. FRS forms a list of the computers corresponding to the members in the member list. All members have an **frsComputerReference** attribute pointing to the computer object.
5. FRS finds the DNS name of each of the computers in the computer list. This is the name that FRS uses to bind to its inbound and outbound partners.

### How Registry Polling Works

FRS polls Active Directory and the registry at the same interval. Some settings stored in Active Directory are also stored in the registry, but the Active Directory settings override the registry settings. Though some registry changes are detected and applied during polling, other changes are not applied until the FRS service is stopped and restarted. For more information about registry entries that require the service to be restarted, see Registry Reference in Tools and Settings.

### Polling Intervals

Each time a replica member is started or the FRS service is started, FRS polls the registry and the computer object in Active Directory in eight short intervals. If no changes in Active Directory are detected, FRS polls in long intervals until it detects configuration or subscriber list changes in Active Directory. The lengths of the short and long intervals are as follows:

- For domain controllers, the default short and long polling intervals are five minutes each. (Domain controllers always use the short interval, regardless of the long interval setting.)
- For member servers, the default short polling interval is five minutes and the long polling interval is 60 minutes.

**Note**

- These polling intervals are specified in the **DS Polling Long Interval in Minutes** and **DS Polling Short Interval in Minutes** registry values. To view the current interval and the length of the long and short polling interval, use the Ntfrsutl.exe command with the **poll** parameter.

If there have been no FRS configuration changes in Active Directory after the completion of eight short polling intervals, FRS automatically begins polling according to the long polling interval. If one of the following events occurs, the polling cycle is reset:

- A replica member is added or removed.
- A connection is added or removed.
- A schedule is changed.
- A file or folder filter is changed.

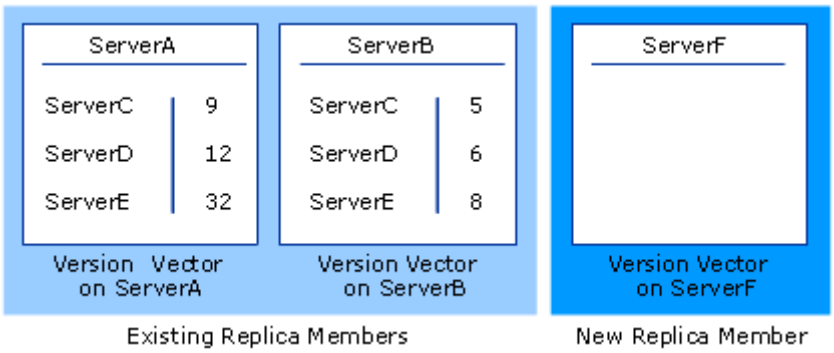
Unlike changes in Active Directory, registry changes do not affect the polling cycle.

### How the Vvjoin Process Works

A version vector is an array that contains pairings of the member originator GUID of the originating member and the VSN of the change. The VSN is a monotonically increasing sequence number assigned to each change that originates on a given replica member. Although the VSN is similar to the USN, FRS does not store the actual USN of the change in the version vector because a replica set can be moved to a different volume, or the volume could be formatted and the replica tree restored from backup. In these cases, the USN could be reset.

The following graphic illustrates a version vector on three servers, ServerA, ServerB, and ServerC.

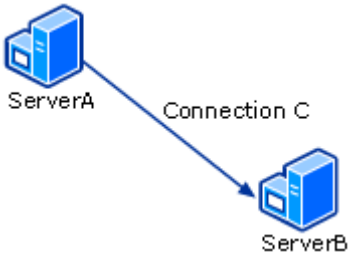
**Example of Three Version Vectors**




Each entry in the version vector is considered the high water mark as to the most recent change order received from the corresponding originator. For example, the version vector on ServerA indicates that this member has received and processed all changes up to VSN 9 originating on ServerC, VSN 12 originating on ServerD, and VSN 32 originating on ServerE. ServerB's version vector shows that its replica tree is not as up-to-date as the replica tree on ServerA. The new member, ServerF, has an empty version vector.

A version vector join (vvjoin) is the process by which a downstream partner joins with an upstream partner for the first time. During this process, the two members compare version vectors to determine which files need to be replicated from the upstream partner to the downstream partner. Vvjoins can occur in a number of scenarios, which are explained using the following figure as an example.

**Sample Replica Members for Vvjoin Scenarios**




In this figure, ServerA and ServerB are members of a replica set. ServerA is upstream and ServerB is downstream for connection C. Using this figure as an example, a vvjoin can occur in the following scenarios:

**Scenario A** Connection C is new, or connection C is deleted and a new connection is created in its place.

**Scenario B** Connection C is disabled and later enabled.

**Scenario C** ServerA polled a domain controller that has not seen the NTDS Connection object for connection C yet. At this point ServerA acts as though the connection is deleted. Later the NTDS Connection object for connection C replicates to this domain controller and ServerA reads it. The effect is similar to Scenario B.

**Scenario D** ServerB is a newly added member of the replica set.

**Scenario E** ServerA or ServerB performs a nonauthoritative restore (D2).

**Scenario F** ServerA performs an authoritative restore (D4), and ServerB performs a nonauthoritative restore (D4).

**Scenario G** ServerB is offline for longer than the **Outlog Change History In Minutes** registry entry, which is 10080 minutes (seven days) by default.

Vvjoins take place when the schedule first opens between the two members in the previous scenarios. The type of vvjoins performed in these scenarios can be either full or optimized vvjoins. These vvjoins are explained in the following sections.

### When Optimized Vvjoins Occur

In scenarios A, B, C, and D, FRS attempts to perform an optimized vvjoin instead of a full vvjoin. An optimized vvjoin occurs when the upstream partner's outbound log contains all the changes that the downstream partner is missing.

Optimized vvjoins are desirable in scenarios where a downstream partner (ServerB in the previous figure) needs to vvjoin and has almost all files in the replica tree already. For example, assume the connection between ServerA and ServerB is new, but ServerB was offline for two days and only missed changes from a prior upstream partner, say ServerC, for those two days. In this case, the new upstream partner (ServerA) retains the change orders in the outbound log for seven days and sends its outbound log to the downstream partner (ServerB) instead of enumerating its entire database. ServerA also sends its existing staging files, so ServerA needs to regenerate staging files only for staging files that were purged if the staging folder reached 90 percent of its limit or that were purged during the staging folder cleanup process. The staging folder cleanup process is described in "How the Outbound Log Cleanup Process Works" later in this section.

### When Full Vvjoins Occur

If the upstream partner's outbound log does not contain all the changes that the downstream partner is missing, a full vvjoin occurs. A full vvjoin requires the upstream partner to enumerate its entire database before determining whether to send changed files and folders to the downstream partner. As a result, the full vvjoin process is not as efficient as an optimized vvjoin, but a full vvjoin does not necessarily mean that the upstream partner sends the entire replica across the network to the downstream partner. The following sections describe the scenarios in which full vvjoins are performed and whether all or part of the data is replicated.

### When a new replica member is prestaged and added to the replica set

In scenario D, the new member (ServerB) has been prestaged with a restored version of the replica tree. When you prestage files, you use a backup program to back up the replica tree and restore it on the new member. Using a backup program is required to preserve the file GUIDs and security descriptors.

Although it might seem that an optimized vvjoin would work in this scenario, optimized vvjoins do not look for prestaged content. Therefore, you must either clear the outbound log of the direct upstream partners or wait until the replica set is at least seven days old (or the duration specified in the **Outlog Change History In Minutes** registry value). Either of these steps will trigger a full vvjoin, which looks for prestaged content and replicates to the new member only the files and folders that have changed since the new member was prestaged. If these steps are not followed, an optimized vvjoin is performed instead, causing all the data in the replica to be replicated across the network to the new member.

**Note**

- The **Outlog Change History In Minutes** registry entry does not take effect until the next polling interval or until you use the **Ntfrsutl poll /now** command.

For more information about prestaging, see "How Prestaging Works" later in this section.

### When a new replica member is not prestaged and the replica set is more than seven days old

In scenario D, if the new member (ServerB) has not been prestaged with a restored copy of the replica tree, and the replica set is more than seven days old (or older than the duration specified in the **Outlog Change History In Minutes** registry value), ServerA enumerates its entire database, sends directed change orders for every file and folder in the replica tree, generates staging files for every file in the replica tree, and then sends all files in the replica tree across the network. In this scenario, the full vvjoin requires a lot of CPU time, disk I/O, and network bandwidth.

### An existing replica member was offline for more than seven days

In scenario G, if a server rejoins a replica set after being offline for more than seven days (or longer than the duration specified in the **Outlog Change History In Minutes** registry value), the upstream partner enumerates its database but only sends across the network files and folders that have changed since the server was offline.

For more information about what happens when a server is offline for more than seven days, see "How the Outbound Log Cleanup Process Works" later in this section.

### How the Partner Join Process Works

The partner join process is a handshake between two existing partners before they begin replicating. The process begins after any of the following events:

- The connection schedule opens.
- The FRS service is started while the schedule is open.
- A connection is reestablished after the loss of network connectivity or server shutdown while the schedule is open.

During the partner join process, which is similar to a server message block (SMB) session setup, the partners negotiate capability level, exchange version vectors, and test for compression support. The schedule must be open for a partner join to occur. On a given connection, the upstream partner has the leading (next change) and trailing (last acknowledged) indexes and calculates what files need to be replicated to the downstream partner.

### What Happens When You Create a DFS Replica Set

After you use the Configure Replication Wizard in the Distributed File System snap-in to configure replication for DFS link targets, the following events take place:

1. FRS-related Active Directory objects and their attributes are created on the server acting as the primary domain controller (PDC) emulator master. (Note that these objects will eventually replicate to all domain controllers.)
2. The FRS service is started on all members and set to automatic, if the members are not currently part of another replica set. The File Replication service event log is also created the first time the service starts.

3. Active Directory is immediately polled after the FRS service starts. If the FRS service was already running, the poll takes place according to the current polling interval. (After FRS detects a change, it begins the eight short polling intervals as described in "How FRS Polling Works" earlier in this section.)
4. If the Active Directory objects have replicated to the domain controller that FRS polls, FRS verifies that the replica root is a valid path, verifies that the staging directory is a valid path (and creates it if necessary), and checks the database path and creates the database. If the Active Directory objects are not yet present on the domain controller that FRS polls, there is a delay in this process until the Active Directory objects are replicated to this domain controller.

The final steps of the process differ for the initial master and the remaining replica members.

**For initial master:**

1. On the initial master, FRS walks the replica tree, adds the files and folders to the file ID table, and stamps a file GUID on every file and folder.
2. FRS declares itself as online and sends a join command to the other members of the replica set. This confirms that FRS is ready to replicate.

**For the remaining replica members:**

1. The remaining replica members come online in the seeding state, which indicates that they are ready to seed (replicate in) the data from an online server.
2. Eventually the remaining replica members vvjoin with other online members.

### What Happens When SYSVOL is Created During Domain Controller Promotion

The SYSVOL shared folder is built by the Active Directory Installation Wizard (Dcpromo) during the installation of Active Directory. The process is as follows:

1. Dcpromo calls FRS to prepare for promotion. If FRS is already running on the server to be promoted, Dcpromo stops the FRS service.
2. Dcpromo deletes information from previous demotion or promotions (primarily FRS-related registry keys).
3. The Net Logon service stops sharing the SYSVOL shared folder (if it exists), and the **SysvolReady** registry entry is set to 0 (false).
4. Dcpromo creates the SYSVOL folder and the necessary subfolders and junction points.
5. The FRS service is started.
6. Dcpromo makes a call to FRS to start a promotion thread that sets the necessary registry keys.
7. Dcpromo reboots the server.
8. When the server is restarted, FRS detects that the server is a domain controller and then checks the registry for the **SysVol Information is Committed** registry entry. Because this entry is set to 0 (false), FRS creates the necessary Active Directory objects and then populates information from the registry to the Active Directory objects and creates the reference attributes as necessary.
9. FRS begins to source the SYSVOL content from the computer that is identified in the **Replica Set Parent** registry entry. (This key is temporary and is deleted after SYSVOL has been successfully replicated.) The connection to the server specified in Replica Set Parent is also temporary. This connection, called a volatile connection, is used to perform the initial vvjoin so that the new domain controller does not need to rely on Active Directory replication for the new connection objects to be replicated.
10. When SYSVOL is finished replicating, FRS sets the **SysvolReady** registry entry to 1 (true), and then the Net Logon service shares the SYSVOL folder and publishes the computer as a domain controller.

### How Prestaging Works

Prestaging is the process of restoring a version of the replica tree on a new server before adding it to a replica set, thus avoiding the need to send the replica tree across the network. Prestaging works for both DFS and SYSVOL replica sets, though the process is slightly different:

- Prestaging a DFS replica member requires you to restore a backup of the replica tree on a server before you add the server to an existing replica set.
- Prestaging a domain controller requires you to restore a system state backup (which includes the SYSVOL shared folder) to be used as the data source when promoting a server running Windows Server 2003 to a domain controller.

### How Prestaging DFS Replica Sets Works

The following procedure describes how FRS responds when you prestage a DFS replica set.

1. Replication is enabled between two members, Server1 and Server2. The initial master, Server1, does not have any data in its replica tree when replication is enabled.
2. After you copy files into the replica tree (using \\Server1\Apps as an example replica root), FRS generates staging files and sends change orders to the downstream partner (Server2). FRS also generates an MD5 (a hash algorithm) checksum during the staging file generation and saves the result in the file ID table on Server1 and in the change order sent to Server2.
3. When Server2 processes this change order, it saves the MD5 checksum in the file ID table on Server2. This process is the only way an MD5 checksum is saved in the file ID table and the presence of the MD5 is necessary to avoid overhead when new members are added later.

   When step 3 is finished, the replicated files exist on both Server1 and Server2, and both file ID tables have MD5 checksums for each file and folder in the replica tree. (You can verify this by using the Ntfrsutil.exe command with the **idtable** parameter. Search for "MD5CheckSum" in the output.)
4. Windows Backup or another backup program is used to back up the contents of the replica tree from either Server1 or Server2. The backup is then restored to another server, for example, Server3.
5. If fewer than seven days have passed since the replica set containing Server1 and Server2 was created, the outbound log must be cleared so that a full vvjoin is triggered when the next member joins. (Setting the **Outlog Change History In Minutes** registry entry to 0 clears the outbound log.)
6. When Server3 is added to the replica set using the Distributed File System snap-in, FRS on Server3 moves all files from the restored target folder (\\Server3\Apps) to the pre-existing data folder, and then initiates a full vvjoin from each computer that Server3 has inbound NTDS Connection objects. Note that the vvjoin process is serialized, meaning that the downstream partner performs a vvjoin with one upstream member at a time. Subsequent vvjoins are much faster because the downstream partner will typically have a complete (or nearly complete) replica tree after the first vvjoin completes.

   The key requirement in this step is that Server3 has inbound connections from an upstream partner, Server1 or Server2 in this case, whose file ID table contains MD5 checksums for files contained in the replica set of interest.
7. FRS on Server1 enumerates all the files and folders in its file ID table and sends directed (that is, single target) change orders to Server3. Because the file ID table has an MD5 checksum, it is included in the change order. As Server3 processes these change orders, this server takes the file GUID for the file or folder from the change order and attempts to locate the corresponding file in the pre-existing data folder. If the server locates the file, it re-computes the MD5 checksum on the content of that file, compares the result to the MD5 checksum it received in the change order and, if they match, uses the pre-existing file instead of attempting to obtain the file from Server1. If Server3 does not find the file, or if the MD5 checksum or attributes do not match, the server obtains the file from Server1. Any change to the file content, such as to the access control lists or data streams, can cause an MD5 mismatch and the file is obtained from Server1 or other upstream partner.

   When all replication activity has settled out, the file ID tables on all three servers have an identical MD5 checksum and identical file content in the replicated folder.

### How Prestaging SYSVOL Works

The process for prestaging SYSVOL is part of the overall process of using backup media to create additional domain controllers. FRS follows the same method of determining whether the restored files are identical to those on the upstream partner as it does for DFS replica members. Specifically, the following requirements must be met for SYSVOL prestaging to be successful:

- FRS must have constructed MD5 checksum data for the files in the SYSVOL tree. MD5 checksum data is constructed after one of the following events occurs:
  - Every file in SYSVOL is modified after there are two or more domain controllers in the domain.
  - All data is moved to a non-replicated folder outside of SYSVOL and then moved back into SYSVOL after there are two or more domain controllers in the domain.
- The system state backup must contain all files in SYSVOL. If any files in the system state backup are out of date, those files are replicated across the network to the new member.
- The outbound log on the upstream partner must be cleared if SYSVOL is less than seven days old. As described in "How the Vvjoin Process Works" earlier in this section, if the replica set is less than seven days old, FRS attempts an optimized vvjoin, which does not look for prestaged content and will cause the entire replica tree to be replicated across the network to the new member. To work around this, you can clear the outbound log on all upstream partners, or you can clear the outbound log on one upstream partner and then do one of the following steps to make sure that the new member vvjoins with this partner:
  - Create a single inbound connection to the member with the cleared outbound log.
  - Set up connection schedules such that the parent with the cleared outbound log is the only schedule that is open.

- Stop the FRS service on all other upstream partners.

**Caution**

- When prestaging SYSVOL, do not clear the outbound log on bridgehead servers. Bridgehead servers typically have many outbound connections, and clearing the outbound log can cause full vvjoins to occur for every connection that has pending outbound change orders. If this occurs, the bridgehead server can experience a significant decrease in FRS performance and significantly increased CPU, memory, and bandwidth usage. To avoid this situation, we recommend that you prestage from an upstream replica member that has few outbound connections.

### What Happens When You Remove a Member from a Replica Set

When a member is removed from a replica set, certain events occur both on the removed member and in Active Directory. The following sections describe what happens when a domain controller is demoted (and thus removed from the SYSVOL replica set) and what happens when you remove a member from a DFS replica set.

### Demoting a Domain Controller

Dcpromo (via Ntfrsapi.dll) performs the following steps during a domain controller demotion:

1. Dcpromo prepares for demotion by:
   a. Stopping FRS.
   b. Cleaning out old demotion state in the registry (if any).
   c. Restarting FRS.
   d. Binding to Active Directory using a different domain controller in the domain. The binding occurs because the demotion will invalidate the local domain controller and thus Active Directory replication would not take place after the system restarts. (If this is the last domain controller in the domain then this step is not performed.)
2. Dcpromo starts the demotion by:
   a. Setting the SYSVOL **Replica Set Command** registry entry to Delete.
   b. Instructing FRS to tombstone the SYSVOL replica set.
3. Dcpromo commits the demotion by:
   a. Stopping the FRS service and setting the service to manual.
   b. Setting the **Sysvol Ready** registry entry to 0.
   c. Deleting the NTFRS Subscription object and NTFRS Replica Set object for SYSVOL.

### Removing a Member from a DFS Replica Set

The following process occurs when you use the Distributed File System snap-in to remove a replica member from a replica set.

1. If the topology is hub and spoke and the hub member is removed, the topology type is changed to custom.
2. The NTFRS Subscriber object and its container are deleted.
3. The connections are adjusted based on current topology preference. For example, if the topology is a ring topology, a ring topology is reformed using the remaining members.
4. The connections from the removed member to other members are deleted.
5. The NTFRS Member object is deleted.

### Types of Changes That Trigger Replication

Because a file is a basic unit of replication, any change to a file will cause the entire file to be replicated after the handle to the file is closed. (Replication takes place either immediately or when the replication schedule opens.) The following table provides a comprehensive description of all changes that can cause a file to be replicated. This table includes the code logged in the FRS outbound logs to indicate the type of change that caused replication to occur. These changes must be followed by a close record in the USN journal before replication begins.

**Types of Changes That Trigger Replication**

| Change Code in Outbound Log | Explanation |
|---|---|
| Create | A file or folder was created. |
| Delete | A file or folder was deleted. |

| | |
|---|---|
| RenNew | A file or folder was renamed. (This type of change also applies to file or folders deleted locally using Windows Explorer, in which case the file or folder is renamed and moved to the Recycle Bin.) |
| DatOvrWrt | Main file data stream was overwritten. |
| DatExt | Main file data stream was extended. |
| DatTrunc | Main file data stream was truncated. |
| Info | Basic information was changed (hidden and read-only attributes, last write time, and so on) Note that some types of basic info changes, such as archive flag or the last access time, do not cause files to be replicated. |
| Oid | Object ID change. |
| StreamNam | Alternate data stream name change. |
| StrmOvrWrt | Alternate data stream was overwritten. |
| StrmExt | Alternate data stream was extended. |
| StrmTrunc | Alternate data stream was truncated. |
| EAChg | Extended file attribute was changed. |
| Security | File permissions, auditing, or ownership were changed. |
| CompressChg | File compression attribute changed. |
| EncryptChg | File encryption changed. Note this type of change triggers replication only when the file is changed from encrypted to not encrypted. |
| Reparse | Reparse point changed. |

The following changes do not trigger replication:

- Changes to a file or folder's last access time
- Changes to a file or folder's archive bit
- Changes to encrypted files
- Changes related to reparse points

The process that FRS uses to determine whether a file has changed is as follows:

1. FRS monitors the USN journal for changes. When FRS detects a close record for a file (a handle being closed will trigger this), FRS gathers relevant information about the recently closed file, including the file's attributes and MD5 checksum, from the file ID table.
2. FRS computes an MD5 checksum for the recently closed file. The MD5 checksum is calculated based on the file's data, including its security descriptors. File attributes are not included in this calculation. If the MD5 checksum and attributes of the recently closed file are identical to the information about the file stored in the file ID table, the file is not replicated. If either the MD5 checksum or the attributes are different, FRS begins the change order process described in "How Files and Folders Are Replicated" later in this section.

Certain types of programs can make identical updates a file without actually changing the file's content (in other words, the file's MD5 checksum for the last change is identical to the previous change to that same file). These programs typically include backup, defragmentation, and antivirus programs that were not written to be compatible with FRS. Setting file system policy on files by using Group Policy can also cause frequent identical updates that trigger replication. These types of updates include:

- Overwriting a file with a copy of the same file.
- Setting the same ACLs on a file multiple times.
- Restoring an identical copy of the file over an existing one.

FRS suppresses excessive replication of files caused by these types of updates and logs FRS event ID 13567 in the File Replication service event log. More specifically, this event is logged when FRS detects that 15 identical updates were made to FRS replicated files within a one-hour period and this condition has occurred over three consecutive hours. The duplicate changes might have occurred on a single file 15 times, or 15 unique files one time each, or any combination between those two.

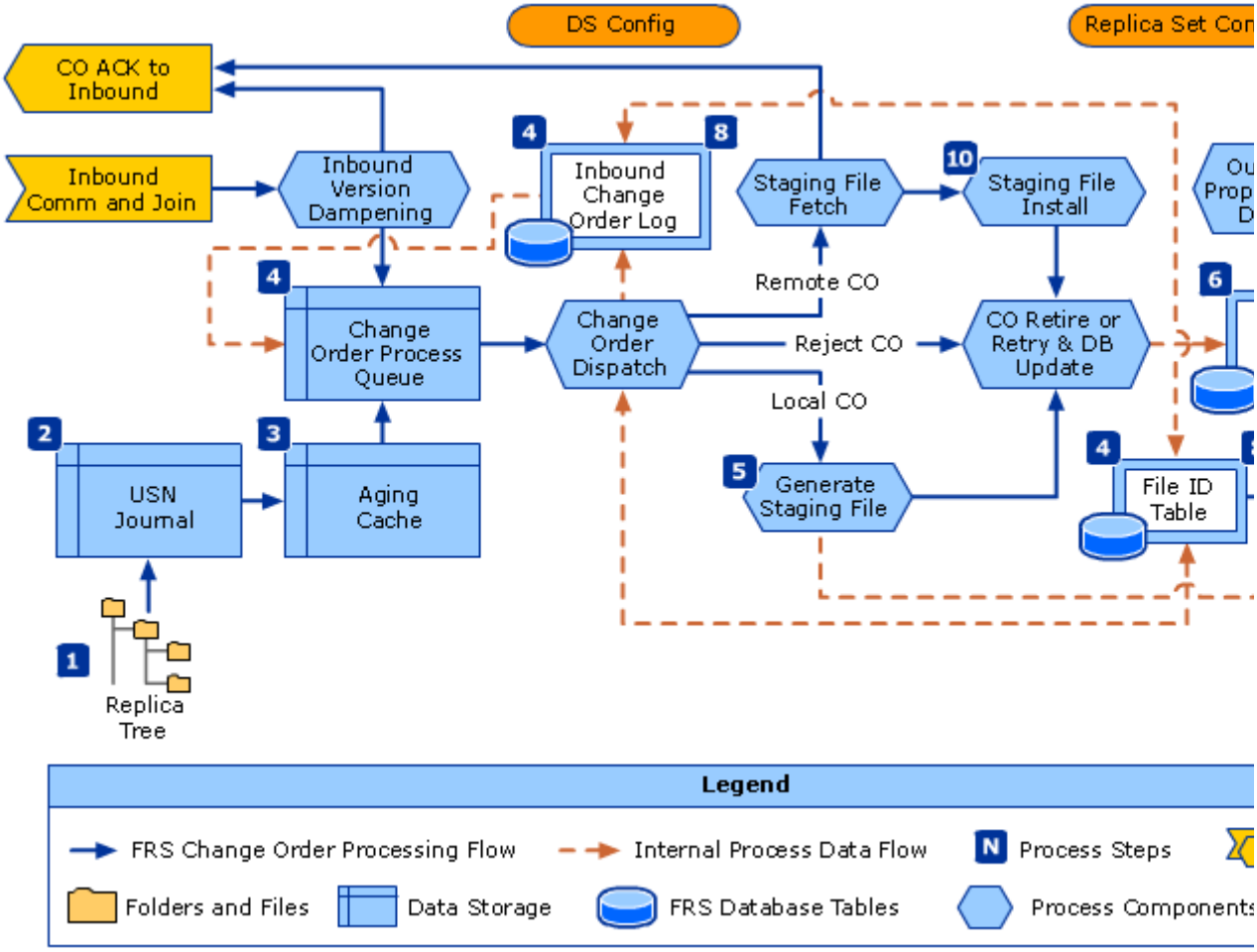
Note that these thresholds apply only to event logging; FRS suppression works continuously. In addition, FRS suppression does not apply to folders. Therefore, frequent updates to folders affect replication and staging even when suppression is turned on.

### How Files and Folders Are Replicated

The following figure describes the change order flow process, which begins when a file or folder is added to a

replica tree or when a file or folder is changed and then closed in a replica tree. The sections that follow describe each step in the process. The differences in the flow process for file deletions are also described.

**How Change Orders Are Processed**




### Step 1: NTFS creates a change journal entry

When a file in a replica set is changed and closed, NTFS makes an entry in the USN journal. The USN journal records changes to all files on the NTFS volume (such as file creations, deletions, and modifications). In Windows Server 2003, the default journal size is 128 MB (or 512 MB in the pre-SP1 release of Ntfrs.exe) and is persistent across restarts and crashes. If the FRS service is stopped or fails, this will have no effect on the replication of FRS content. Even with the service stopped, file changes are still recorded at the file system level in the USN journal. When the FRS service is restarted, stored USN journal entries will trigger replication.

### Step 2: FRS monitors the USN journal

FRS monitors the USN journal for changes that apply to the replica trees on ServerA. Only closed files are checked. File and folder filters are applied against changes in the files and folders in the replica tree.

### Step 3: Aging cache is used

The aging cache is a three-second delay designed to catch additional changes to a file. This prevents replication when a file is undergoing rapid updates.

### Step 4: FRS creates entries in the inbound log and file ID table

ServerA records the change as a change order in its inbound log. It also creates an entry in the file ID table so that recovery can take place if a crash occurs. The inbound log contains change orders arriving from all inbound partners. Change orders are logged in the order that they arrive. Each change order contains information about a change to a file or folder on a replica member, such as the name of the file or the time it was changed. This information is used to construct a message about the change.

### Step 5: FRS creates the staging file in the staging folder

ServerA uses backup APIs to create a backup of the changed file or folder in ServerA's staging folder. These backup files, known as staging files, encapsulate the data and attributes associated with a replicated file or folder. By creating the staging file in the staging folder, FRS ensures that file data can be supplied to partners regardless of any activity that might prevent access to the original file. All staging files in the staging folder are

compressed to save disk space and network bandwidth during replication.

At step 1 if a file is deleted, FRS does not create a staging file.

#### Step 6: FRS creates the entry in the outbound log

ServerA updates the outbound log. The outbound log contains change orders generated for a specified replica set. These change orders can originate locally or come from an inbound partner. Change orders recorded in the outbound log are eventually sent to all outbound partners.

If a file is deleted, the change order contains the event time of the deletion and the tombstone lifetime (by default 60 days). When the tombstone for the deleted file expires, the file's entry in the file ID table is purged.

#### Step 7: FRS sends a change notification

ServerA sends a change notification to ServerB.

#### Step 8: FRS records and acknowledges the change notification

ServerB stores the change order in its inbound log and file ID table. If ServerB decides to accept the change order, it sends a change order acknowledgement (CO ACK) to ServerA to replicate the modified file's staging file.

#### Step 9: FRS replicates the staging file

ServerB copies the staging file from ServerA to the staging folder on ServerB. ServerB then writes to its outbound log so other outbound partners can pick up the change.

If the change order is for a deleted file, no staging file is fetched. The change notification for the deleted file is added to ServerB's outbound log,

#### Step 10: FRS constructs the staging file and moves it into the replica tree

ServerB uses restore APIs to reconstruct (that is, restore) the file or folder in the preinstall folder, and then FRS renames the file or folder into the replica tree.

If the change order was for a file deletion, ServerB deletes the file from the replica tree.

### How the Staging Folder Works

The staging folder is a queue for changes to be replicated to downstream partners. After the changes are made to a file and the file is closed, the file content is compressed, written to the staging folder, and replicated according to schedule. Any further use of that file does not prevent FRS from replicating the staging file to other members.

The following sections describe various aspects of the staging folder.

#### Staging folder size

The size of the staging folder governs the maximum amount of disk space that FRS can use to hold those staging files and the maximum file size that FRS can replicate. The default size of the staging folder is approximately 660 MB, the minimum size is 10 MB, and the maximum size is 2 terabytes. The largest file that FRS can replicate is determined by the staging folder size on both the upstream partner and downstream partners and whether the compressed replicated file can be accommodated by the current staging folder size. Therefore, the largest file that FRS can replicate is 2 terabytes, assuming that the staging folder size is set to the maximum on upstream and downstream partners.

#### Staging folder compression

FRS replica members running Windows 2000 Server with Service Pack 3 (SP3) or later or Windows Server 2003 compress the files replicated among them. Compression reduces the size of files in the staging folder on the upstream partners, over the network between compression-enabled partners, and in the staging folder of downstream partners prior to files being moved into their final location.

All files and subfolders within the staging folder are compressed. However, compression is not enabled on the staging folder itself.

#### How the staging folder stays below its maximum size limit

When FRS tries to allocate space for a staging file and is not successful because the size of the files in the staging folder has reached 90 percent of the staging folder size, FRS starts to remove files from the staging folder. Staged files are removed (in the order of the longest time since the last access) until the size of the staging folder has dropped below 60 percent of the staging folder limit. Additionally, staging files for downstream partners that have been inaccessible for more than seven days are deleted. As a result, FRS does not stop replicating if the staging folder runs out of free space. This means that if a downstream partner goes offline for an extended period of time, the offline member does not cause the upstream partner's staging folder to fill with accumulated staging files.

**Note**

- During the process of deleting staging files to reach the 60 percent limit, FRS does not delete staging files that have change orders in the inbound log pending install, because they are still needed by this member.

The fact that FRS removes files from the staging folder does not mean that the underlying file is deleted or will not be replicated. The change order in the outbound log still exists, and the file will eventually be sent to downstream partners when they process the change order. However, before replication can take place, the upstream partner must recreate the staging file in the staging folder, which can affect performance. Recreating the staging file can also cause a replication delay if the file on the upstream partner is in use, preventing FRS from creating the staging file.

### Staging folder cleanup

Staging files are cleaned up during the outbound log cleanup process. Specifically, staging files are deleted from the staging folder after all replica members have acknowledged the change orders that generated the staging files. For more information about the outbound log cleanup process and its effect on the staging folder, see "How the Outbound Log Cleanup Process Works" later in this section.

### How Replication Schedules Work

The replication schedule defines the periods during which replication takes place. The following sections describe various aspects of replication schedules.

### SYSVOL Schedules

The SYSVOL schedule is an attribute associated with each NTFRS Replica Set object and with each NTDS Connection object. FRS replicates SYSVOL using the same intrasite connection objects and schedule built by the KCC for Active Directory replication. FRS uses two replication protocols for SYSVOL:

- **SYSVOL connection within a site**. The connection is always considered to be on; any schedule is ignored and changed files are replicated immediately.
- **SYSVOL connection between sites**. SYSVOL replication is initiated between two intersite members at the start of the 15-minute interval, assuming the schedule is open. The connection is treated as a trigger schedule. The upstream partner ignores its schedule and responds to any request by the downstream partner. When the schedule closes, the upstream partner unjoins the connection only after the current contents of the outbound log, at the time of join, have been sent and acknowledged.

The SYSVOL replication schedule can be configured so that replication takes place four, two, one, or zero times per hour.

- When set to four times per hour, replication starts at 0:00, 0:15, 0:30, and 0:45. This schedule is essentially continuous replication.
- When set to two times per hour, replication occurs at 15-minute intervals starting at 0:15 and 0:45.
- When set to one time per hour, replication starts at 0:00 and ends at 0:15 assuming that all changes have been replicated.

It is possible to change the SYSVOL schedule, but this should be done only after careful consideration of the implications and alternatives. Also, note that schedules exist on both site links and NTDS Connection objects. If you change the schedule on an NTDS Connection objects, the connection then becomes a manual connection that cannot be managed by KCC.

### DFS Replica Set Schedules

For DFS replica sets, FRS uses the NTDS Connection objects, topology, and schedule built by the Distributed File System snap-in. The snap-in sets the schedule as follows:

- When you view the properties of a link and set the schedule in the ***Linkname*** **Properties** dialog box, the snap-in applies the schedule to all NTDS Connection objects for this replica set. The snap-in does not apply the schedule to the NTFRS Replica Set object.
- If you view individual connections in the **Customize Topology** dialog box and modify the schedule on a connection, the snap-in applies that schedule to the NTDS Connection object for the connection.
- If you add a new member to an existing replica set, the new member's NTDS Connection objects get the default "always on" schedule.
- If you use the snap-in to set schedules on individual connections, and then you view the schedule from the ***Linkname*** **Properties** dialog box, the snap-in displays the schedule for the last enumerated NTDS Connection object. If you click **OK** in this dialog box, the schedule shown is applied to all NTDS Connection objects. To avoid this, click **Cancel** to close the dialog box.

When replication is set to available, replication occurs continuously. Unlike SYSVOL schedules, DFS replica set schedules are on/off schedules. As long as the schedule is open, the connection stays in the joined state and

change orders are sent out as they are processed by the upstream partner. Any queued change orders in the upstream partner's outbound log are sent first. When the schedule closes, FRS unjoins the connection.

### How Local Time Affects the Schedule

Replication schedules are stored in Coordinated Universal Time (UTC). However, when you view the schedule, the schedule is shown in the local time of the server. Daylight savings time causes the schedule to shift by an hour for any server that is located in an affected time zone.

### How File and Folder Filters Work

Filters exclude subfolders (and their contents) or files from replication. You exclude subfolders by specifying their name, and you exclude files by using wildcard characters to specify file names and extensions. By default, no subfolders are excluded. The default file filters exclude the following files from replication:

- File names starting with a tilde (~) character
- Files with .bak or .tmp extensions

Filtering takes place when FRS monitors the USN journal for changed files and folders. If a new or updated file meets a filter that excludes it from replication, FRS ignores the change in the USN journal and does not generate a change order.

Because filtering occurs at the USN journal, filters act as exclusion filters only for new files and folders added to a replica set. They have no effect on existing files in the replica set. For example, if you change the existing file filter from "*.tmp, *.bak" to "*.old, *.bak," FRS does not go through the replica set and exclude all files that match *.old, nor does it go through the replica set and begin to replicate all files that match *.tmp. After the filter change, new files matching *.old that are added to the replica set are not replicated. New files matching *.tmp that are added to the replica set are replicated.

Any file in the replica set that was excluded from replication under the old file filter (such as Test.tmp, created when the old filter was in force) is automatically replicated under the new file filter only after the file is modified. Likewise, changes to any file that was not excluded from replication under the old filter (such as Test.old, created when the old filter was in force) continue to replicate under the new filter, until you explicitly delete the file.

These rules apply in the same manner to the folder exclusion filter. If a folder is excluded, all subfolders and files under that folder are also excluded.

Regardless of the filters you set, FRS always excludes the following from replication:

- NTFS mounted drives.
- Files encrypted by using EFS.
- Any reparse points except those associated with DFS namespaces. If a file has a reparse point used for Hierarchical Storage Management (HSM) or Single Instance Store (SIS), FRS replicates the underlying file but not the reparse point.
- Files on which the temporary attribute has been set.

### How the Outbound Log Cleanup Process Works

The outbound log cleanup process runs periodically to remove change orders from the outbound log. The process is scheduled by the outlog thread, and the first cleanup is scheduled to run 60 seconds after the outlog thread starts. Service initialization code starts the outlog thread.

At every scheduled cleanup, the time it took to perform the last cleanup is calculated and the next scheduled time is set. The time before the next scheduled cleanup is set to 50 multiplied by the time it took to run the last cleanup. This interval is then adjusted so that it is not less than 60 seconds, it is not more than the **Outlog Change History In Minutes** registry entry (which is seven days, by default), and it is not more than eight hours. As a result, the cleanup process tries not to take more than 2 percent of the processing time. For large outbound logs, the cleanup can take several minutes to run.

The cleanup process does not need to run every time it is scheduled. The cleanup process only runs for the replica sets that have received at least one change order acknowledgement (ACK) since the last time cleanup ran on that replica set. In situations where a replica member does not receive ACKs from any partners for a long time, there is an interval after which the cleanup process runs on that replica member, even if it has not received an ACK from its partners. The interval is either the amount specified in the **Outlog Change History In Minutes** registry entry (seven days, by default) or eight hours, whichever is smaller.

The cleanup function goes through all the change orders in the outbound log and does the following.

1. Deletes all vvjoin change orders and their corresponding staging files if they have been acknowledged by the partner they were meant for. (Vvjoin change orders are deleted because they are directed change orders, which means they are generated for a specific partner and are of no use after they are acknowledged by that partner.)
2. Deletes staging files for all the change orders that have been acknowledged by all the partners. Staging files are not kept longer than required; only change orders are kept. Staging files are

regenerated as needed.

3. Deletes change orders that have been in the outbound log for more than the **Outlog Change History In Minutes** registry entry. If there is a connection that has not yet sent the change order that is being deleted, that connection is marked for deletion at the next poll. The deletion of these connections will take them out of the list of active connections and trigger cleanup on the change orders that were backed up for them. Because these connections are not actually deleted from Active Directory, they will reappear at the second poll and will rejoin and go through a complete vvjoin. This functionality provides a way for FRS to clean up change orders in the outbound log and staging files for members that have not accepted changes for a long time.

### How FRS Debug Logs Work

FRS creates text-based logs in the *systemroot*\Debug folder to help you debug problems. The Ntfrs log files store transaction and event detail in sequentially numbered files (Ntfrs_0001.log through Ntfrs_0005.log). Transactions and events are written to the log with the highest sequence number in existence at that time. The characteristics of the log files are determined by the values of several registry entries in the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters subkey. After the number of logs specified by the value of the **Debug Log Files** registry entry has been filled, the lowest log version is deleted and the remaining log file names are decreased by *n*–1 to make room for a new log file.

Log detail is controlled by the value of the **Debug Log Severity** registry entry, ranging from 0 to 5, with 5 providing the most detail. Log size is determined by the value of the **Debug Maximum Log Messages** registry entry. The default value of 20,000 lines for **Debug Maximum Log Messages** results in a log file of approximately 2 MB, for a total of 10 MB of logs (**Debug Log Files** * [**Debug Maximum Log Message** * 120]). Setting **Debug Log Messages** to 50,000 results in log files of approximately 5 MB and 25 MB of total log space with default settings.

Solving problems by using the Ntfrs logs requires ensuring that the value of the **Debug Log Severity** registry entry is set high enough to capture the events needed to identify the problem. Severity settings range from 0 to 5 and are cumulative, meaning that a setting of 4 includes log events with a severity of 0 to 3. Error logs hold a severity setting of 0 and are always displayed. Setting the severity level at 5 will cause every action to be logged and can make finding more important information difficult. The default log setting is 2.

All log records have a date and time stamp and an identifying string annotated by **::** with a letter or pair of letters between each colon. These identifying strings make it easy for you to identify errors, warning messages, and milestone events in the log files and filter out less important information. For example, a log entry that has a **:U:** as an identifying string includes information related to the USN journal. A log entry containing information about Active Directory polling will have an identifying string of **:DS:**. A tracking log entry has an identifying string of **:T:** and summarizes a change order that has finished or is in the process of updating a given file or folder. The following table lists the log record identifiers.

**Log Record Identifiers**

| Identifier | Description |
|---|---|
| :: | Change order trace records |
| ++ | Continuation records |
| :DS: | DS access entries |
| :FK: | FRS registry Key entries |
| :H: | Log header entries |
| :S: | Service startup and shutdown entries |
| :SC: | Service controller entries |
| :SR: | Send / Receive command server entries |
| :T: | Tracking record |
| :U: | USN journal entries |
| :V: | Version Vector join entries |
| :X: | Communication entries |

Tracking record entries (**:T:**) can be a helpful way to identify and understand problems that can occur during the change order process. A tracking record entry will tell you what files have been changed and where the change originated.

The following tracking record entry describes a remote change order that is creating a new file called test_file in Replica-A. The version number is zero.

```
7/31-08:40:08 :T: CoG: d42cda60 CxtG: 000001b7 [RemCo   ] Name: test_file
7/31-08:40:08 :T: EventTime: Mon Jul 31, 2003 08:40:04 Ver: 0
7/31-08:40:08 :T: FileG: ceff96a6-5c9f-433a-989c841454a1593b FID: 61a70000 0000036c
```

```
7/31-08:40:08 :T: ParentG: 1a89f4e1-a0c0-43e4-aedbe869f767f372 Size: 00000000 0000000
7/31-08:40:08 :T: OrigG: 2eea81b4-f92d-4941-9f269d4bbdd7ea05   Attr: 00000020
7/31-08:40:08 :T: LocnCmd: Create State: IBCO_COMMIT_STARTED          ReplicaName: Re
7/31-08:40:08 :T: CoFlags: 0000040c   [Content Locn NewFile ]
7/31-08:40:08 :T: UsnReason: 00000002 [DatExt ]
```

The individual fields that comprise the tracking log entry are described in the following table. In some cases only the first DWORD of a GUID is actually displayed in the log.

**Tracking Log Entries**

| Identifier | Description |
|---|---|
| :T: | Identifying string. |
| CoG: | Change order GUID - Uniquely identifies a create/delete/rename/modify action for a file. |
| CxtG: | Connection GUID - Identifies the connection object in the topology connecting an upstream member to the member that delivered this change order. |
| [ ] - RemCo | Identifies a remote change order. |
| [ ] - RemCo, Abort | Identifies a remote changer order that was aborted. |
| [ ] - LclCo | Identifies a local change order. |
| [ ] LclCo, Abort | Identifies a local change order that was aborted. |
| Name: | File name. |
| EventTime: | Time on the originating member at which the change was performed. |
| Ver: | Version number of the file. Increases by one each time a local change order is created. |
| FileG: | File GUID - Uniquely identifies the file or folder and is used as the NTFS object ID on the file or folder. The corresponding file/folder on each replica member has the same file GUID. |
| FID: | File ID - The NTFS volume-specific file ID (file reference number). |
| ParentG: | Parent GUID - The GUID of the parent folder that contains this file or folder. |
| Size: | The approximate size of the file or folder noted in hexadecimal. |
| OrigG: | Originator GUID - The GUID associated with the member of the replica set that originated this update. |
| Attr: | File attributes - The attribute flags for the file/folder. |
| LocnCmd: | Location command - One of Create, Delete, NoCmd, MoveDir; indicating that the file is being created, deleted, updated, or is changing parent folders. |
| State: | The change order state - One of IBCO_STAGING_RETRY, IBCO_FETCH_RETRY, IBCO_INSTALL_RETRY, IBCO_COMMIT_STARTED, IBCO_INSTALL_REN_RETRY, IBCO_INSTALL_DEL_RETRY, and IBCO_ FETCH_REQUESTED indicating that the change order is being retried later because of inability to complete the fetch of the staging file, or inability to install the change to the file. Finished change orders have a state of IBCO_COMMIT_STARTED. |
| ReplicaName: | The name of the replica set containing this file or folder. |
| CoFlags: | Change 0rder flags<br>• Abort - Set when CO is being aborted.<br>• VVAct - Set when VV activate request is made.<br>• Content - Valid content command.<br>• Locn - Valid location command.<br>• LclCo- CO is locally generated.<br>• Retry - CO needs to retry.<br>• InstallInc - Local install not completed.<br>• Refresh - CO is an upstream-originated file refresh request.<br>• OofOrd - Don’t check/update version vector.<br>• NewFile - If CO fails, delete IDTable entry.<br>• DirectedCo - This CO is directed to a single connection.<br>• DemandRef - CO is a downstream demand for refresh. |

- VVjoinToOri - CO is from vvjoin to originator.
- MorphGen - CO generated as part of name morph resolution.
- MoveinGen - This CO was generated as part of a sub-dir MOVEIN.
- OidReset - All CO did was reset the object identifier back to FRS defined value.
- CmpresStage - The stage file for this CO is compressed.

UsnReason: Flags set in the NTFS change log describing modifications to the file.

- Close - Change log close record.
- Create - File or folder was created.
- Delete - File or folder was deleted.
- RenNew - File or folder was renamed.
- DatOvrWrt - Main file data stream was overwritten.
- DatExt - Main file data stream was extended.
- DatTrunc - Main file data stream was truncated.
- Info - Basic info change (attrib, last write time, and so forth).
- Oid - Object ID change.
- StreamNam - Alternate data stream name change.
- StrmOvrWrt - Alternate data stream was overwritten.
- StrmExt - Alternate data stream was extended.
- StrmTrunc - Alternate data stream was truncated.
- EAChg - Extended file attribute was changed.
- Security - File access permissions changed.
- IndexableChg - File change requires reindexing.
- Hlink - Hard link change.
- CompressChg - File compression attribute changed.
- EncryptChg - File encryption changed.
- Reparse - Reparse point changed.

The following tracking log entry describes a local change order (a change order originating on the computer where the log was produced) that is updating the same file, test_file. The version number is now 1. Notice that the originator GUID is different from that of the tracking log entry above. The file GUID and parent GUID of both log entries are the same for both change orders because the same file is involved and it has not changed parent directories.

```
7/31-08:56:55 :T: CoG: cd55ad6f CxtG: 37b12c93  [LclCo ]  Name: test_file
7/31-08:56:55 :T: EventTime: Mon Jul 31, 2003 08:56:52   Ver:  1
7/31-08:56:55 :T: FileG: ceff96a6-5c9f-433a-989c841454a1593b FID:  61a70000 0000036c
7/31-08:56:55 :T: ParentG: 1a89f4e1-a0c0-43e4-aedbe869f767f372 Size: 00000000 0000020
7/31-08:56:55 :T: OrigG: 8f759ded-e611-43c4-be05c10138dfdea4 Attr: 00000020
7/31-08:56:55 :T: LocnCmd:   NoCmd    State: IBCO_COMMIT_STARTED      ReplicaName: Re
7/31-08:56:55 :T: CoFlags:   00000024   [Content LclCo ]
7/31-08:56:55 :T: UsnReason: 00000002   [DatExt ]
```

---

Back to Top

## Network Ports Used by FRS

FRS uses the following two network ports:

**Network Ports Used by FRS**

| Service Name | UDP | TCP |
|---|---|---|
| LDAP | | 389 |

| | |
|---|---|
| RPC | Dynamic |

By default, FRS replication over remote procedure calls (RPCs) occurs dynamically over an available port by using RPC Endpoint Mapper (also known as Remote Procedure Call Server Service or RPCSS) on port 135; the process is the same for Active Directory replication.

---

Back to Top

### Related Information

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

- Active Directory Replication Topology Technical Reference
- Volume Shadow Copy Service Technical Reference