Distributed Directory Services

Avinash Gupta Tharun Mohandoss Sayan Mandal

March 19, 2018

1 Introduction

What is directory service?
Directory services are software systems that store information relating to users and other organisational objects in an organised way and also provide methods to access them to users.

Why is it used?
The directory services have several advantages mentioned below:

- Centralised user account management
- Centralised policy management (group policy)
- Better security management
- Replication of information between DC's

2 X.500 Directory Standard

The Directory
The directory stores information that is strictly relevant to an entry in a concise way. It is just like a database but is more descriptive and the data it holds is attribute based.

Directory Model
The requestors of directory services include both users and computer processes. The end users can be diverse but the model makes them uniform by directory user agent(DUA). DUA is a computer process which provides an interface to the user and acts as a directory user to access the directory. The DUA includes functions to access the directory.The aim of the directory model is to provide all users with the same information if they have access to it.

Directory Information Base
Directory Information Base (DIB) is the metadata about objects being stored in the directory. It has an interface between the users and system providers.

The Distributed Directory
The X.500 standard was designed to allow the directory to be distributed among more than one systems which together will form the complete directory.Each of these systems is called a DSA(Directory System Agent).The directory is a set of these DSAs. Each DSA holds a fragment of the DIB and responds directly to requests for information in its fragment. But the DSAs can also get requests for the information which is held by some other DSA. So, in order to respond to these requests the DSA needs to know some information about others fragment which is called glue. The glue helps in routing the request to the DSA which holds the information. There are two modes of routing: Informing the requester about the DSA which holds the desired information . This approach is known as referral. The requester can request the referred DSA for information. The other mode is to forward the request to the corresponding DSA which is known as chaining. The DSA responds back with the information to the requesting DSA which in turn sends the information to the requestor.

Directory Security
The directory contains widely available information which is distributed among numerous systems. The providers of this information might not intend this information to be shared to all the users and hence the information needs to be controlled. The controlling of information arises the need for security which including user authentication and access rights to information.
For user authentication, the directory generally uses an authentication protocol. But for access control, the requirements vary widely and depends on the policies of the organisation which owns the corresponding information. Some part of the information can be available to everyone while some part of it can be protected and available to users which have the required rights.In the 1992 edition, a basic access control protocol was defined.

Entries and attributes
Each entry is composed of information about a single object. Each entry consists of a set of attributes. Each attribute stores some information about the object. An attribute has an attribute type and value. An attribute can also have multiple values. All the attributes of an entry have distinct types.
Every attribute type has a unique object identifier. The aim of object identifier is to avoid any confusion about the attribute type. The object identifier is a sequence of non negative numbers. The telephone number attribute is allocated 2 5 4 20 as the object identifier.

Subtype
The 1992 edition also has the concept of attribute type hierarchies. Using the hierarchy, a new attribute type(subtype) can be defined from another attribute type(supertype). The subtype attribute value is indirectly associated with the supertype which allows the user to get information at different levels of specificity.

Object Classes
An object class is a a collection of similar objects which share certain characteristics. Every object class has mandatory and optional set of attribute types. The entry for each object that belongs to the object class must contain an attribute corresponding to each mandatory type whereas those in optional set need not be there.

Hierarchy in Object Classes
We can define a new object class as a subclass of another class(superclass). The mandatory set of subclass must contain all the attribute type present in the mandatory set of the superclass. An attribute type in the optional set of superclass can be promoted to mandatory set in the superclass but all the remaining attribute types in the optional set of superclass must be present in optional set of subclass. New attribute types can be added to both the mandatory and optional set of subclass.

Abstract Class
An abstract class cannot have any objects directly associated with it. It can only be used to define other object classes as a subtype of it.

Structural Object Class
It is used to define the structure of the DIB and provides attribute types to generate the names of objects.

Auxiliary Object Class An auxiliary object class is used to add attributes to another object class. It is useful when a set of attribute types is used by multiple classes.

Matching Rules
A matching rule is used to compare objects with the information provided by the user to select the matched objects for further scrutiny.

Directory Information Tree
The DIB is stored as a hierarchical structure which is a tree and hence called directory information tree (DIT) . Each node of this tree is an entry. The DIT provides a unique and well defined name to every entry. This name includes the RDNs of all the nodes in the path from root to the entry node including itself. The directory also includes aliases which violate the pure tree structure. These alias entries point to the original object and hence logically the object now has more than 1 parent.

Directory Schema
There are some built-in rules in the directory itself which it enforces on objects like allowing only a single attribute of any attribute type within an object. Other rules need to be defined by an organisation or some standard group.The directory schema is the set of defined rules. When a change is made to the DIB, the directory enforces the directory schema upon it. The rules in directory schema correspond to attribute types, object classes and matching rules.

DIT Structure And Content Rule
A DIT Structure Rule is applied to a specific object class in some part of DIT and specifies which superior structure rules and name forms to be applied to objects of that class.
A DIT Content Rule governs the content of entries by dictating which auxiliary objects to use to form entries of a particular structural class.It is also applied in some part of DIT.

Naming Authorities and RDN
The DIT ensures that subordinates of every entry has unique relative distinguished name (RDN).

The naming authority has to ensure that RDNs are distinct. Every object with a non leaf entry has a naming authority. The naming authority registers the names and the directory publishes them.It assigns a unique RDN to every object which is a subordinate of a particular entry.

Distinguished Names Each object has a distinct name. The distinguished name of an object consists of RDNs all the superiors starting from root and ending with the objects RDN. As DIT is a tree, every objects entry has a unique path from the root and since every entry with the same immediate superior has a distinct RDN, the final distinguished name is unique.

Aliases
An object can have one or more alias names. An alias is also a path from root the root to an entry. Any name for an alias entry can also be used for the entry to which the alias points to.

3 LDAP : Lightweight Directory Access Protocol

Introduction
LDAP or lightweight directory access protocol is a protocol that aims to provide access to directories supporting X.500 models but it does not require as much resources as the Directory Access Protocol.

Protocol model
In this protocol, whenever the client needs to perform an operation, it transmits a request to the server and the server sends either the results(success) or the error that occured. In some cases, the server may return a referral to another server.

Data Model
One or more server is assumed to provide access to the directory which can be hierarchically arranged in the form of a tree. Each node in the tree represents a attribute value pair that is relatively distinguished(no two siblings have same attribute value pair). Distinguished Name of a node is defined as the collection of relative distinguished names of the nodes when following the path from the root of the DIT to the node itself.

Attributes
Each entry contains attributes i.e a set of types and associated values. Each attribute type has a descriptive name and an OID. Each attribute type contains details about the syntax/conditions that a value has to satisfy for it to be valid and comparison rules(to check equality, greater than, less than etc.).

Object Classes
Each entry also has a structural object class which denotes the kind of object that the entry represents and may have other auxiliary classes which help specify other characteristics for the entry. These also have object identifiers(OID). Each object class can have a set of compulsory/optional attribute list that any entry of that type must/may have.

Schema
An LDAP schema consists of the following elements:

1. **Attribute syntaxes**
   This consists of information that define what values are valid for a particular value type. For example a name might be defined to be a string of alphabets(no numerals or special characters).

2. **Matching rules**
   Matching rules define a logical way of comparing two elements of LDAP data. There are three basic types of matching rules.
   (a) **Equality** : To check if two values are equal.
   (b) **Ordering** : To check if one value is greater-than/less-than etc to another value. This is used to order values.
   (c) **Substring** : To check if one value is a substring of another
   Other types of matching rules may be present such as approximate matching rules etc.

3. **Attribute types**
   This contains definitions of the different types of attributes. An attribute type defines properties about how clients and server should interact with an attribute of this type.

4. **Object classes**

5. **Naming Forms**
   Naming attributes are the attributes that appear in an entry's RDN. Name forms can be used to specify a set of required and optional attributes.

6. **DIT Content rules**
   The set of attributes that are required and optional for an entry are defined in the object class. DIT content rules further add to this and can impose extra constaints on entries having a particular structural object class.

7. **DIT Structure rules**
   DIT structure rules impose restriction on where an entry can be placed in the DIT based on the hierarchical relationships between entries.

Operations
The interaction between client and server happens in the form of 'operations' performed by the client on the directory via the server. All messages from client to server can be described as a operation request and all the messages from the server to client can be described as responses to these operations requests(Except 'Unsolicited messages'). The different types of operations are:

- **Bind Operation**
  Before the client can start performing other operations on the directory, it must first 'bind' to a server creating a 'protocol session'. This can be done through a bind request which has

authentication information as well the LDAP version supported/expected by the client.

- **Unbind operation**
  This request terminates the current protocol session.

- **Search operation**
  Through this operation a client can try to find entries that satisfy certain criteria. The important elements of a search operation are:
  1. **Base Object :** The entry relative to which the search takes place.
  2. **Scope :** The client can choose to search either at just the base object or it's children or the whole subtree rooted at the base object.
  3. **Filter :** The particular criteria that a entry has to satisfy to qualify as a valid result. This must be specified by the client. Some of the common filers are Equality of an attribute, presence of an attribute, substring etc. Even complex and custom matching rules can be defined.

- **Modify operation**
  There are four types of modify operations:
  1. Add : Add a value to an attribute
  2. Delete : Remove a value from an attribute
  3. Replace : Replace the list of attribute values
  4. Modify DN: Changes the distinguished-name of the entry which may result in change in it's RDN or relocation of the subtree rooted a the entry.

  The user must have proper authentication and also the modification must result in a consistent state, otherwise the operation fails.
- **Add operation**
  This operation is used to request addition of an entry to the Directory. This operation may also require authentication.

- **Delete operation**
  This operation is used to request removal of an entry from the Directory. This operation may also require authentication.

- **Compare operation**
  This operation is used to request comparison of an assertion to an entry in the Directory.

- **Abandon operation**
  The client can request the server to abandon a previously requested operation. Even if a the client sends this, this may not ensure prevention of the operation requested earlier from happening. The only thing that this ensures is that the server definitely does not send a response to the previous operation after receiving a abandon request for that operation.

- **Extended operation**
  This helps clients request custom operations that are not available otherwise in the protocol.

Referral
In LDAP, the server can sometimes send a Referral as a response instead of success or explicit failure. The referral is sent when the server could not successfully process the requested operation but has reason to believe that the same request if performed to another server may result in success. For example, if a client makes a read request to an entry that is present in a different server, the server may send the URL of the server that it thinks contains the entry as a response.

Unsolicited notification
An unsolicited notification from the server to a client as the name suggests is not incited by any operation request. Rather it is unsolicited in the sense that it is sent from server to client not in response to any request. It is generally used to signal some out of the ordinary situation that the client might need to be aware of.

Security
Security is extremely important in case of directories. This is especially true when dealing with sensitive information that should ideally be accessible only to authorized individuals. In LDAP, authentication is achieved during the bind operation where the client sends his/her authentication information. The important ways of authentication are:

1. **No Authentication**
   The simplest one. This is used when data security is not of concern.

2. **Simple Authentication**
   In this case, the DN and the password are sent by the client in plaintext or Base64 encoded format. Although more secure than No authentication, the encryption is not hard to break if the data is captured between in the network.

3. **Simple Authentication and Security Layer**
   SASL is a framework for adding additional authentication mechanisms to connection-oriented protocols

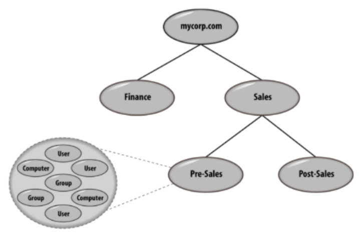
4 Design Issues

- **Naming of entries**
  Every entry needs to be provided a unique name.
- **Replication of data**
  The data needs to be replicated to make the system fault tolerant.It is also required to identify which data to replicate and how.
- **Distribution of directory and DIT**
  The DIT needs to be distributed among DSA's and they need to respond to requests for information held by other DSAs.
- **Consistency Maintenance**
  The directory needs to be kept consistent and various conflicts needs to be resolved.

5 Microsoft Active Directory

5.1 Introduction

Active Directory is a central repository for management of users, resources in an Enterprise. Microsoft Active Directory is Microsofts Network Operating system originally built on windows 2000. It provides a view of single repository to administer the Enterprise although the directory is globally distributed. Most windows server operating systems include Active Directory as a set of processes. The server running the domain service is known as Domain Controller. It controls the authorization and access to the resources in the domain. For example, when a user tries to log into an computer it authorizes. Data in the active directory is stored in a tree and each entry is referred as Object. As in a tree, there are internal node which is called container and leaf which is called non-container.
This is an example of a domain. To access an object, we have to use Distinguished name according to LDAP standard. Distinguished names are used according to syntax provided in the LDAP standards. Distinguished name in a domain for an object must be unique. Relative Distinguished name (RDN) is used to uniquely identify an object within its parent container. Distinguished name for an object consists of concatenating the RDN of its parents upto the root. RDN of two objects can be same but they cannot be under the same container, they can be under different container. RDN of siblings cannot be same. For example, from the above picture if all of the containers were organizational unit, then the RDN for Pre-Sales and Post-Sales would be: ou=Pre-Sales,ou=Sales,dc=mycorp,dc=com ou=Post-Sales,ou=Sales,dc=mycorp,dc=com respectively.



Domain and Domain Trees: Domain constructs the logical structure of Microsoft Active Directory.Domain Controller can be in charge of only one domain only. Domain can be defined as the group of network resources that share the same active directory database. For example, in the above picture mycorp has been assigned the mycorp.com as DNS domain name. So it has been implicitly defined as the root of the hierarchical structure which is called as the Domain Tree. If the company later decides to extend its organization to asia,europe and africa, the domain name for them will be mycorp.com.asia, mycorp.com.europe, mycorp.com.africa respectively and they will be children of the mycorp.com.All the domains inside a domain tree implies transitive trust. This implies if A trusts B and B trusts C, A trusts C implicitly. Parent-child trust is implicit. For example, administrator in asia.mycorp.com can allow any user in europe.mycorp.com to access any resource in asia.mycorp.com.

Forests: Forest consists of one or more domain trees. The domain trees in a forest share common global catalog, directory schema and configuration. and connected together through transitive trust. Forest is named after the first domain tree created.

Organizational Unit: Objects within a domain can be grouped into an Organizational Unit(OU). An organizational unit can apply group policies to the objects within it. User accounts in the different OU within a domain cannot be same. For example, sam.market-ou.com and sam.sales-ou.com cannot exist. OU can contain hierarchy of container and objects.

Global Catalog: Global catalog in a forest is used to implement forest wide search as it contains the subset of attributes of each object in the forest. Any search for an object needs to first go through the global catalog and then redirect it into the corresponding domain controller.

FSMO ROLES: In a multi-master based model, such as Active Directory updates can happen in any domain controller. But this can introduce the update conflict during the replication. Although there is certain rule like Last write wins which is applied by windows, for certain kind of objects it is better to avoid this conflict rather than try to solve it. For this reason, windows performs updates to certain objects in a single-master model where a single domain controller is assigned the role of a master and update can happen to only master and it includes multiple roles which include:

Schema Master(Forest-wide): in charge of changing the directory schema.

Domain naming master(Forest-wide): in charge of adding and removing domains in the forest and renaming and moving any domain in a forest.

RID master(Domain-wide):when a domain controller acts an user or a group, then a SID and RID is attached to the object which is unique for each object. So when the available RID falls below a threshold, the domain controller requests for RID to it and it allocates the RID pool to domain controller.

PDC master(Domain-wide): For the backward-compatibility, one domain controller has to act as a PDC master in a domain. One of its main job is for PDC chaining, whenever a domain controller changes password for some account, it forwards the information to the PDC. next time if any account wants to authenticate and the local domain controller doesnt think the password is correct it forwards the request to the PDC. One other main job is it acts as a primary time server for the whole domain.

Infrastructure master(Domain-wide): when an object in one domain is referenced by an object in another domain it is referenced by its GUID, SID and DN. The primary role of the infrastructure master is to update the SID and DN in a cross-referencing domain object. The server which is acting as a global catalog cannot act as an infrastructure master because global catalog contains subset of attributes from each object, it will stop updating the object.

5.2 Search Operation

Active Directory search is based on a client and server model. To do a search operation, client constructs a request based on LDAP syntax and sends to the active directory which includes bind, add, modify, unbind. Directory system agent which runs on every domain controller receives the request and processes it.To search in an active directory client has to follow four steps:

- Finding LDAP server
- Establish connection
- Authenticate against it
- Perform the search

In addition to it, it may include LDAP referrals in which case an LDAP server redirects the request to another domain controller which may contain the queried object.

Finding LDAP server: When a domain controller first starts up, it first registers its service register number in the DNS. So when a client searches for the LDAP server it queries in DNS for hosts which provides directory services.

Establish Connection: To establish the connection, the LDAP client starts a TCP session on the port on which the LDAP server is listening to and then connect on that port.

Authentication: The domain controller provides password to provide authentication to the client.

Performing Search Operation: Performing a search produces all the objects in the corresponding search scope which matches the criteria. To perform a search client needs to mention the search scope, search base, selection and the filter. Search base includes the location of the object where the search will begin. Search scope includes the base search, one-level search, subtree search to denote how deep will the search go from the base. Filter includes the objects which meets the criteria. Selection denotes the required attributes of the matched objects.

5.3 Replication in Active Directory

Active Directory is a Distributed Directory service. Objects in the directory is distributed among the domain controllers in a domain or forest. Updates can be changed to any of the domain controller. The corresponding update should be updated in the domain controller which hold the same copy of the object. Active directory creates a site topology for replication which uses the most efficient link connection among the sites.

Replication model: The Directory tree defines all the objects in the directory. Active directory allows the tree to be partitioned in a way so that it can be distributed to all the domain controllers in a forest. Each domain controller keeps some portion of the tree. Each defined partition of the tree is called a segment. On update, the update is replicated to all the domain controllers which maintains the copy of same directory portion.

Pull Replication: Active Directory uses pull based replication. In this model, the destination requests for the updates from the source and includes the updates it has received from the source and the other domain controller. The source domain controller calculates the updates it needs to send to the destination and sends to the destination. The destination on receiving the updates from the source apply the update and next time it requests to any other domain controller for the updates, it excludes the recent updates that it has applied to itself. The alternative design policy is push based replication, where a source sends its updates to the destination. But it can be problematic as there is no way for the source to know which updates the destination needs. So it may happen that the destination does not apply the updates it receives from the source.

Replication Model Components:

Multi-master, Loose, convergence: The replication is multi-master because there is more than domain controller which stores the same portion of the tree where updates can happen. Loose consistency model because at any point it doesnt guarantee that two replicas will be consistent. Convergence because if it is allowed to reach a stable state where no new updates will not happen and all the previous updates have been replicated successfully, all the replicas will have same value for the same copy. In a single-master replication model, the update can be applied to only one server and it Replicates the update to other backup server. It can be suitable for a small organization, However it is not suitable for a large organization. With Active directory multiple domain controller can act as a master and it need not replicate to all the domain controller so that It doesnt increase the replication traffic in a network and doesnt increase the replication Latency. Single-master model can be disadvantageous because if the primary server fails, Directory updates cannot take place. For example, suppose the primary server fails and One users password has expired so it cannot logon because he cannot reset the password.

Store and Forward replication: Store and Forward replication because an update is not sent to all the domain controllers at once. In stead change in any one domain controller is sent to only a subset of other domain controllers which in turn sends the update to other subset of domain controllers until all the domain controller receives the update.

State-based Replication: Active Directory replication is state-based, which means instead of sending the change logs it sends the updated value of the object to other domain controllers. In a log-based system, the source sends its change logs to all the domain controllers and after receiving the change logs, the domain controller applies the update to keep itself more up-to-date. Each originating writes is assigned a sequence number in the originated replica and each replica maintains information about how updated they are with respect to the other replicas.

The replication strategy taken by active directory has some advantages because:

- This strategy is able to distinguish between the objects that has been deleted and a new object which has the same distinguished name as the deleted one because the replication is not based on DN, it is based on GUID which is unique for each object created.
- Only the attributes of an object, not the whole object is transmitted as a part of replication. So it can resolve the write conflicts.
- The replication strategy involves efficient network communication between the replicas, which in return reduces the network latency.

Directory Partition Replicas: A directory partition replica can be either full or partial. Full replica means the replica contains all the attributes of all the objects of the partition and read-writable, whereas the partial means only a subset of attributes. Each domain controller keeps at least three full partition replica which contains schema partition, which stores the information about class and attributes across the whole forest. Configuration partition, which stores the replication configuration information same for all domains. Domain Partition, which stores the objects in a domain. Domain partition is replicated in all the domain controllers within a domain. A partial replica contains subset of attributes of all partition objects and it is stored in Global Catalog Servers.

Active Directory Updates: When a User updates an object in a domain controller, it notifies other domain controllers within sites about the change and other domain controller request for the updates. For inter site, it does an scheduled replication.

Originating Updates: An LDAP server can add an object, modify an object, move an object and delete an object from the directory. An LDAP server processes each write request as an atomic operation, even if one update of attribute of the object fails, the whole transaction fails and object is not modified.

Multi-master Conflict Resolution Policy: There are three write conflicts that can happen in multi-master LDAP directory server:

•**Attribute value:**
A modify operation changes the value of an attribute. Concurrently, another domain controller sets the same attribute to a different value.

•**Add or Move under deleted parent:**
An add or move operation may make C as a child of P. Concurrently, on a different domain controller delete operation deletes P.

•**Sibling name Conflict:**
concurrently on different domain controller an operation creates two different objects with same relative distinguished name.

The policy taken by active directory to resolve these conflicts is to order every operation by assigning a globally unique GUID to the originating update and resolve based on the order of the GUID of the update. For example, for the attribute value conflict, after resolution all replica contains the attribute value whose GUID is larger. After resolution of the second conflict, P is deleted from all replicas and C is made child of some special Lost And Found object container in the domain partition. After the resolution of the third conflict, it assigns a unique relative distinguished name to the object with lower timestamp so that it cannot conflict.

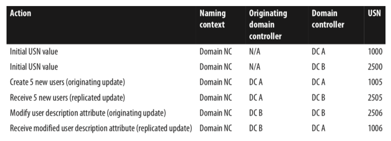
Replicating the Conflict Resolution: Lets say that server A requests replication from server B and receives updates from server B. next it requests replication from server C and receives updates from server C. When applying these updates in order, it detects the conflict between these updates and resolves the conflict according to the above policy and finds in server Cs favour. So, B needs to know the update from C. To do this, when B requests update from A, it gets the update from C and when applying it finds the conflict and resolves the conflict and finds in Cs favour. So, B finally have Cs value. Additional problem may occur when changes to server occur and the server goes down prior to replication. If the server never comes back, those changes are obviously lost. For the following subsection, we will assume a topology where server A,B and C are interconnected.
Replication metadata: Replication metadata is the information about the data being replicated that active directory stores to govern replication between domain controllers without ended up being in a loop or missing any data. The replication data for schema naming context is held in schema naming context, and is separated from configuration naming context which is held in configuration

naming context.

Update Sequence number and HighestCommittedUSN: Each domain controller assigns a Unique Sequence Number(USN) to each of its atomic transaction. Each time an update operation occurs, a transaction occurs. A USN is a 64-bit value that is assigned to each update transaction on a domain controller. Each separate transaction will generate an incrementing USN for the domain controller. A USN can denote an update of all the attributes of an object or a single attribute of an object. If HighestCommittedUSN for a domain controller is 1000, and 20 minutes later it is 1056, that means 56 transaction occurred in this domain controller in the last 20 minutes. USNs are used to uniquely identify the update in a naming context in a domain controller. It is highly improbable that same USN will denote the same update in two different domain controller. For example, the update of joeys computer name to Joey may be denoted by USN 123 in one computer and USN 456 in another computer.

Example: If we create 5 users on DC A its USN will be incremented by 5 and when it replicates to DC B, DC Bs USN will be incremented by 5. Similarly if DC A receives a change from DC B, its USN will be incremented by 1. Its shown in the table below:

High-Watermark Vector: The highest watermark vector table is a Table maintained independently by each domain controller for each naming context. The table contains entry for each of the domain controller it is replicating to.
There is one table for each naming context that the domain controller is maintaining a replica of, so there is at least 3 table in each domain controller, for naming context, configuration context and domain context. Each table stores the highest USN it has received from each of its partner it is replicating to, so that it requests for the most recent updates in next replication cycle. When a domain controller initiates a update request to a domain controller, it sends the HWMV value for that domain controller. The destination domain controller then compares the HWMV value and its highest committed USN to decide which updates to send to the source domain controller.



Action	Naming context	Originating domain controller	Domain controller	USN
Initial USN value	Domain NC	N/A	DC A	1000
Initial USN value	Domain NC	N/A	DC B	2500
Create 5 new users (originating update)	Domain NC	DC A	DC A	1005
Receive 5 new users (replicated update)	Domain NC	DC A	DC B	2505
Modify user description attribute (originating update)	Domain NC	DC B	DC B	2506
Receive modified user description attribute (replicated update)	Domain NC	DC B	DC A	1006

Up-to-dateness vector: Up-To-Dateness vector table is a Table maintained independently by each domain controller for tracking originating updates that are received from all source domain controllers to guide efficient replication and to stop needless replication this reducing replication traffic in the network. When a domain controller sends replication request for a naming context it sends its up-to-dateness vector and HWMV to the source domain controller. Source domain controller then compares to filter out the changes that the destination has already received from other domain controllers to eliminate the duplicate replication. This is called propagation Dampening.
Once DC C receives update from DC A, it will try to replicate to DC B. However, DC B has already received those updates from DC A. So DC C will check DC Bs up-to-dateness vector table and will not send those updates to DC B thus preventing an endless loop and unnecessary replication traffic.

Change Notification: How does a DC know when does it need replication? When any update is applied to a DC it notifies its replication partner in a site which is called change notification.

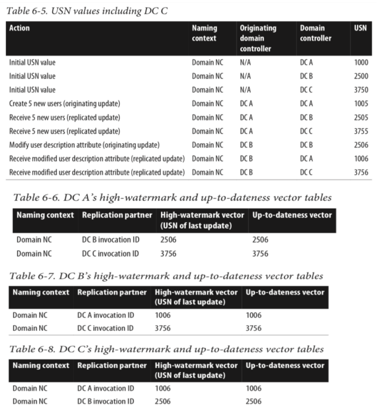


Table 6-5. USN values including DC C

Action	Naming context	Originating domain controller	Domain controller	USN
Initial USN value	Domain NC	N/A	DC A	1000
Initial USN value	Domain NC	N/A	DC B	2500
Initial USN value	Domain NC	N/A	DC C	3750
Create 5 new users (originating update)	Domain NC	DC A	DC A	1005
Receive 5 new users (replicated update)	Domain NC	DC A	DC B	2505
Receive 5 new users (replicated update)	Domain NC	DC A	DC C	3755
Modify user description attribute (originating update)	Domain NC	DC B	DC B	2506
Receive modified user description attribute (replicated update)	Domain NC	DC B	DC A	1006
Receive modified user description attribute (replicated update)	Domain NC	DC B	DC C	3756

Table 6-6. DC A's high-watermark and up-to-dateness vector tables

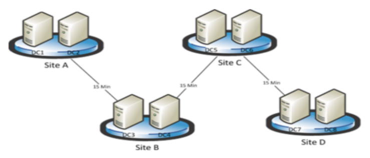
Naming context	Replication partner	High-watermark vector (USN of last update)	Up-to-dateness vector
Domain NC	DC B invocation ID	2506	2506
Domain NC	DC C invocation ID	3756	3756

Table 6-7. DC B's high-watermark and up-to-dateness vector tables

Naming context	Replication partner	High-watermark vector (USN of last update)	Up-to-dateness vector
Domain NC	DC A invocation ID	1006	1006
Domain NC	DC C invocation ID	3756	3756

Table 6-8. DC C's high-watermark and up-to-dateness vector tables

Naming context	Replication partner	High-watermark vector (USN of last update)	Up-to-dateness vector
Domain NC	DC A invocation ID	1006	1006
Domain NC	DC B invocation ID	2506	2506



Change Between Sites: Change between sites happen in a scheduled manner instead of notifying other domain controller. Each domain controller requests for a change to other domain controller in other sites in a timely manner which can be as high as 15 minutes. To converge the replication quickly between two different sites, we have to turn the change notification on between two different sites.

Notification delay: In windows 2003, each DC waits for 15 seconds before notifying its replication partner. So lets consider the example below. This time the inter site change notification is turned on.
What happens when an originating writing occurs at DC1. It waits for 15 second before notifying DC2. When DC2 gets the update it waits for 15 second before notifying DC3 in site B. So this way, by the time it reaches to DC8 in site D the time elapsed is 105 seconds.

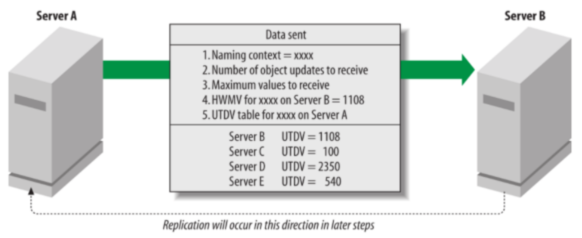
Urgent Replication: Although a convergence rate of 2 min. Is good enough, what happens for account lockout? It may be possible that user connects to domain controller that the lockout has not been replicated yet. So in this case urgent replication helps which achieves almost instantaneous replication to DC8.
Password Changes: This is just on scenario which illustrates the urgent replication. When a user changes the password, it is immediately replicated to the PDC emulator. So when a user tries to logon to a domain controller that the password has not been replicated yet, it requests to the PDC emulator and authenticates.

The Replication of a naming context between two servers: The following examples include server A,B,C,D and E. The replication between any two servers follows the following steps:

- The server initiates update request to the partner.
- Partner works out the updates to send to the source.
- Partner sends the update
- Source applies the update
- Source server checks whether it is up-to-date.

Step 1:

- The name of the NC(Naming Context) it wishes to replicate.
- The maximum number of object updates it wishes to receive from server B.
- The maximum number of values that it wishes to receive during this replication cycle.
- The USN for server B from server As HWMV table.
- Server As up-to-dateness vector for the NC.


Server A
Server B
Data sent
1. Naming context = xxxx
2. Number of object updates to receive
3. Maximum values to receive
4. HWMV for xxxx on Server B = 1108
5. UTDV table for xxxx on Server A
Server B UTDV = 1108
Server C UTDV = 100
Server D UTDV = 2350
Server E UTDV = 540
Replication will occur in this direction in later steps

Step 2:
After receiving the request server B first sees its own Highest Committed USN and compares it to the USN submitted through the HWMV. Any difference in this suggests the potential updates to be sent to A. For example: In this image, server B has 1109,1110,1111,1112 potential updates to be

Server B USN	Originating-DC-GUID	Originating-DC-USN
1109	Server E's GUID	567
1110	Server E's GUID	788
1111	Server B's GUID	1111
1112	Server D's GUID	2345

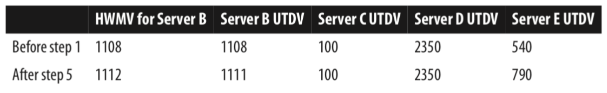
sent to server A as USN through the As HWMV was 1108. But we can see that server Ds UTDV from server A was 2350 whereas in server B this value is 2345, so server A has already got the update from server D. So server B need not send this update to server A.

Step 3:
Server B identifies the list of updates to be sent to A that A hasnt seen from any other sources and sends to A. In addition, if the more-data flag is false, it sends this flag. The returned information from server B includes:

- The list of updates.
- Server Bs last object USN changed.
- The more-data flag.
- Server Bs UTDV for this NC.

Step 4:
Server A now applies each update to the object in its own copy of active directory database and assigns a USN to each of the transaction and modifies its LastUSNChanged after each transaction. After all the transactions have been processed, server A change its HWMV for server B to the last object USN changed, which in this example is 1112. This implies that server A knows of the most recent update in B.

Step 5:
Sever A now checks for the more-data flag, if it is true it goes back to step 1 and requests for replication to server B. If more-data flag is false, it knows that it has received all the updates and it updates its up-to-dateness vector. The up-to-dateness vector helps server A to know which originating updates server B has seen and by replication, it has now seen. For each entry in the UTDV it does two things, if the server is not enlisted in its own UTDV, then it inserts the server in the UTDV table. If the received UTDV is larger than the sent UTDV for the corresponding server, it replaces the UTDV for that server with the received one. Example:



	HWMV for Server B	Server B UTDV	Server C UTDV	Server D UTDV	Server E UTDV
Before step 1	1108	1108	100	2350	540
After step 5	1112	1111	100	2350	790

All images in this document are taken from : Desmond, Brian, et al. Active Directory. O'Reilly Media, 2013.

References:

- Desmond, Brian, et al. Active Directory. O'Reilly Media, 2013.
- https://technet.microsoft.com/en-us/library/cc961788.aspx
- Steedman, Douglas. X.500: the Directory Standard and its application. Technology Appraisals, 1993.
- RFC2251 Lightweight Directory Access Protocol (v3). M. Wahl, T. Howes, S. Kille. December 1997. (Format: TXT=114488 bytes) (Obsoleted by RFC4510, RFC4511, RFC4513, RFC4512) (Updated by RFC3377, RFC3771) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC2251)