AD Fundamentals

mercredi 27 novembre 2024 12:18 PM

@Anis OUERSIGHNI

Active Directory Structure

Active Directory (AD) is a **directory service for Windows network environments**. It is a **distributed**, **hierarchical structure** that **allows for centralized management of an organization's resources**, including users, computers, groups, network devices and file shares, group policies, servers and workstations, and trusts. AD **provides authentication** and **authorization functions** within a Windows domain environment. **A directory service**, such as **Active Directory Domain Services (AD DS)** gives an organization ways to **store directory data** and **make it available** to both **standard users** and **administrators** on the same network. **AD DS stores information** such as **usernames** and **passwords** and **manages the rights needed for authorized users to access this information**. It was first shipped with Windows Server 2000; it has come under increasing attack in recent years. **It is designed to be backward-compatible**, and many features are arguably not "secure by default." It is difficult to manage properly, especially in large environments where it can be easily misconfigured.

Active Directory flaws and misconfigurations can often be used to obtain a foothold (internal access), move laterally and ver tically within a network, and gain unauthorized access to protected resources such as databases, file shares, source code, and more. **AD is essentially a large database accessible to all users within the domain**, **regardless of their privilege level**. **A basic AD user account with no added privileges can be used to enumerate the majority of objects contained within AD**, **including but not limited to:**

Domain Computers	Domain Users
Domain Group Information	Organizational Units (OUs)
Default Domain Policy	Functional Domain Levels
Password Policy	Group Policy Objects (GPOs)
Domain Trusts	Access Control Lists (ACLs)

For this reason, we must understand how Active Directory is set up and the basics of administration before attempting to atta ck it. It's always easier to "break" things if we already know how to build them.

Active Directory is **arranged in a hierarchical tree structure**, **with a forest at the top** containing one or more domains, which can themselves have nested subdomains.

A **forest is the security boundary within which all objects are under administrative control**. A forest **may contain multiple domains**, **and a domain may include further child or sub-domains**.

A **domain is a structure within which contained objects** (users, computers, and groups) **are accessible**. **It has many built-in Organizational Units** (OUs), such as **Domain Controllers**, **Users**, **Computers**, and **new OUs can be created as required**.

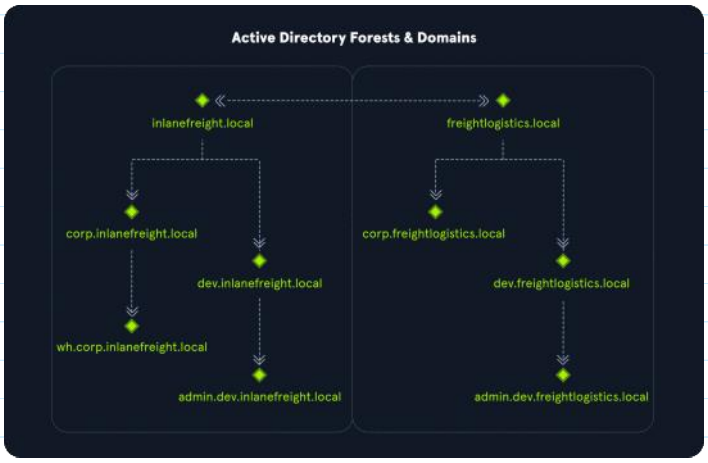
OUs may **contain objects and sub-OUs**, allowing for the assignment of different group policies.

At a very (simplistic) high level, an AD structure may look as follows:

```
INLANEFREIGHT.LOCAL/
|── ADMIN.INLANEFREIGHT.LOCAL
|   |── GPOs
|   └── OU
|       └── EMPLOYEES
|           |── COMPUTERS
|           |
|           |       └── FILE01
|           |
|           |── GROUPS
|           |
|           |       └── HQ Staff
|           |
|           └── USERS
|                   └── barbara.jones
|
|── CORP.INLANEFREIGHT.LOCAL
└── DEV.INLANEFREIGHT.LOCAL
```

Here we could say that **INLANEFREIGHT.LOCAL** is the **root domain** and **contains the subdomains** (either **child** or **tree root domains**) **ADMIN.**INLANEFREIGHT.LOCAL, **CORP.**INLANEFREIGHT.LOCAL, and **DEV.**INLANEFREIGHT.LOCAL as well as **the other objects that make up a**

domain such as **users**, **groups**, **computers**, and **more** as we will see in detail below. **It is common to see multiple domains** (**or forests**) **linked together via trust relationships in organizations that perform a lot of acquisitions.** It is often quicker and easier to create a trust relationship with another domain/forest than recreate all new users in the current domain. As we will see in later modules, domain trusts can introduce a slew of security issues if not appropriately administered.


Active Directory Forests & Domains
inlanefreight.local
freightlogistics.local
corp.inlanefreight.local
dev.inlanefreight.local
wh.corp.inlanefreight.local
admin.dev.inlanefreight.local
corp.freightlogistics.local
dev.freightlogistics.local
admin.dev.freightlogistics.local

The graphic below shows **two forests**, **INLANEFREIGHT.LOCAL** and **FREIGHTLOGISTICS.LOCAL**. The **two-way arrow** represents **a bidirectional trust between the two forests**, meaning that **users in INLANEFREIGHT.LOCAL can access resources in FREIGHTLOGISTICS.LOCAL and vice versa**. We can also see **multiple child domains under each root domain**. In this example, we can see that **the root domain trusts** **each of the child domains**, **but the child domains in forest A do not necessarily have trusts established with the child domains in forest B**. This means that a user that is part of **admin.dev.freightlogistics.local would NOT be able to authenticate to machines in** the **wh.corp.inlanefreight.local** domain **by default even though a bidirectional trust exists between the top-level** inlanefreight.local and freightlogistics.local **domains**.

To allow direct communication from admin.dev.freightlogistics.local and wh.corp.inlanefreight.local, another trust would need to be set up.

Active Directory Terminology

Object

An object can be defined as **ANY resource present within an Active Directory environment** such as **OUs**, **printers**, **users**, **domain controllers**, etc.

Attributes

Every object in Active Directory has an associated set of attributes used to **define characteristics of the given object**. A *computer object* contains **attributes** such as the **hostname** and **DNS name**. **All attributes in AD have an associated LDAP name** that can be **used when performing LDAP queries**, such as displayName for Full Name and given name for First Name.

Schema

The Active Directory schema is essentially **the blueprint of any enterprise environment**. It **defines what types of objects can exist in the AD database and their associated attributes**. It **lists definitions corresponding to AD objects** and **holds information about each object**. For example, users in AD belong to the class "user," and computer objects to "computer," and so on. **Each object** has its own information (some required to be set and others optional) that are stored in Attributes. When an object is created from a class, this is called instantiation, and an object created from a specific class is called an instance of that class. For example, **if we take the computer RDS01. This computer object is an instance of the "computer" class in Active Directory.**

Domain

A domain is a **logical group of objects** such as **computers**, **users**, **OUs**, **groups**, etc. We can think of each domain as **a different** country within the EU. Domains **can operate entirely independently** of one another **or be connected via trust relationships**.
PS : All domains in a single AD forest automatically have implicit transitive trust relationships established between them.

When you create a new domain in an AD forest, it automatically establishes two-way transitive trust with all other domains in the forest.
We can impose trust relationship In cases where you want finer control over access between domains (e.g., not allowing all users from a trusted domain to access resources).

Forest

A forest is a collection of Active Directory domains. It is **the topmost container** and **contains all of the AD objects introduced below**, including but not limited to domains, users, groups, computers, and Group Policy objects. **forest can contain multiple trees**. A forest **can contain one** or **multiple domains**. Each **forest operates independently but may have various trust relationships with other forests**.

- The forest contains a global catalog that allows objects from all domains in all trees to be located and queried.

Tree

A tree is a collection of Active Directory domains **that begins at a single root domain**. A **forest is a collection of AD trees**. Each domain in a tree shares a boundary with the other domains. **A parent-child trust relationship** is formed when a domain is added under another domain in a tree. Two trees in the same forest cannot share a name (namespace). Let's say we have two trees in an AD forest: inlanefreight.local and ilfreight.local. A child domain of the first would be corp.inlanefreight.local while a child domain of the second could be corp.ilfreight.local. All domains in a tree share a standard Global Catalog which contains all information about objects that belong to the tree.

- **Each tree has its own unique namespace** (e.g., tree1.com and tree2.net) **but still belongs to the same forest.**
- **All trees in a forest share the same Active Directory schema**, which defines the structure and rules for object types (e.g., users, groups) and attributes.
- The configuration data, such as site and replication settings, **is shared across all trees in the forest.**

Tree 1 : INLANEFREIGHT.LOCAL and **Tree 2 : FREIGHTLOGISTICS.LOCAL**.
Both trees are part of the same forest and share trust relationships, schema, and configuration data.

Container

Container objects **hold other objects** and **have a defined place in the directory subtree hierarchy**.

Leaf

Leaf objects **do not contain other objects** and are **found at the end of the subtree hierarchy**.

Global Unique Identifier (GUID)

A GUID is **a unique 128-bit value assigned when a domain user or group or generally when Every single object is created**. This GUID value **is unique across the enterprise**, similar to a MAC address. The GUID is **stored in the ObjectGUID attribute**. When querying for an AD object (such as a user, group, computer, domain, domain controller, etc.), **we can query for its objectGUID value** using PowerShell or search for it by specifying its **distinguished name**, **GUID**, **SID**, or **SAM account name**. GUIDs are **used by AD to identify objects internally**. Searching in Active Directory by GUID value is probably the most accurate and reliable way to find the exact object you are looking for, especially if the global catalog may contain similar matches for an object name. Specifying the ObjectGUID value when performing AD enumeration will ensure that we get the most accurate results pertaining to the object we are searching for information about. **The ObjectGUID property never changes and is associated with the object for as long as that object exists in the domain.**

Security principals

Security principals are **anything that the operating system can authenticate**, including **users**, **computer accounts**, or even **threads/processes** that run in the context of a user or computer account (i.e., an application such as Tomcat running in the context of a service account within the domain). In AD, security principals **are domain objects that can manage access to other resources within the domain**. We can also have **local** user accounts and security groups used to **control access to resources on only that specific computer**. These are not managed by AD but rather by the Security Accounts Manager (SAM).
In simple terms, **security principals** are identities in a computer system or network that can be given permissions to access resources. They represent anything that can be authenticated and authorized to do something, like:

1. **Users** - People with usernames and passwords (e.g., JohnDoe or DOMAIN\Jane).
2. **Groups** - Collections of users (e.g., "Admins" or "Employees") that share the same permissions.
3. **Computers** - Machines on the network (e.g., Workstation01$).
4. **Services** - Applications or system processes that run under specific accounts (e.g., SQL Server Service).

5. **Built-in Accounts** - Special accounts like SYSTEM, Administrator, or Guest.

Think of it like this:

If your house is a network, security principals are the people (users), families (groups), and devices (e.g., a robot vacuum or a delivery drone) that you allow inside and give certain permissions—like opening doors, turning on lights, or accessing specific rooms.

Security Identifier (SID)

A security identifier, or SID is **used as a unique identifier for a security principal or security group**. **Every account, group, or process has its own unique SID**, which, in an AD environment, is **issued by the domain controller** and **stored in a secure database**. **A SID can only be used once**. Even if the security principle is deleted, it can never be used again in that environment to identify another user or group. **When a user logs in, the system creates an access token for them which contains the user's SID**, **the rights they have been granted**, and **the SIDs for any groups that the user is a member of. This token is used to check rights whenever the user performs an action on the computer**. **There are also well-known SIDs that are used to identify generic users and groups**. **These are the same across all operating systems**. An example is the Everyone group.

A **SID** (Security Identifier) is like a unique ID number for every user, group, or computer in a Windows system. **It's how the system keeps track of who's who, even if you rename the account.**

Example:

- If your username is "John," your SID might look like this:
  S-1-5-21-3623811015-3361044348-30300820-1013.

The system uses this SID to assign and check permissions, not the name "John." Even if you change the name, the SID stays the same. **It's like your account's fingerprint!**

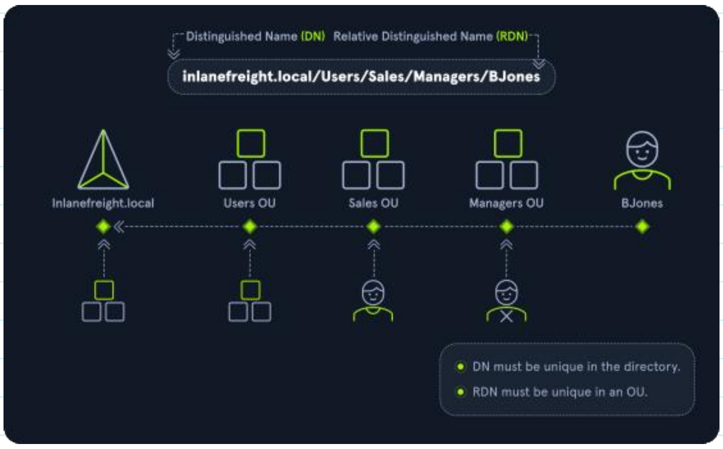
GUID (Globally Unique Identifier):	**SID (Security Identifier):**
1. **Purpose:** ◦ A **permanent, immutable identifier** for an AD object. ◦ Used internally by AD to track objects, even if their attributes (like name) or location within the directory change. 2. **Format:** ◦ 128-bit value represented as a hexadecimal string, e.g., 550e8400-e29b-41d4-a716-446655440000. 3. **Uniqueness:** ◦ Globally unique **across all directories** and **systems**. 4. **Scope of Use:** ◦ Primarily used within the AD database. ◦ Rarely seen or used directly by administrators. 5. **Key Feature:** ◦ Remains unchanged even if the object is moved or renamed. 6. **Example Use Case:** ◦ AD replication relies on GUIDs to identify and synchronize objects across domain controllers.	1. **Purpose:** ◦ A **security-related identifier** for users, groups, computers, or other security principals. ◦ Used in **access control** to determine permissions and rights to resources. 2. **Format:** ◦ Variable-length structure starting with S-1- followed by domain and object-specific identifiers, e.g., S-1-5-21-3623811015-3361044348-30300820-1013. 3. **Uniqueness:** ◦ Unique within a **domain** or **forest**. 4. **Scope of Use:** ◦ Used in ACLs (Access Control Lists) to manage permissions on objects (files, folders, etc.). 5. **Key Feature:** ◦ **Changes if the object is moved to a different domain** (as it becomes associated with a new domain SID). ◦ Includes a **RID (Relative Identifier)** to uniquely identify objects within the same domain. 6. **Example Use Case:** ◦ A file's ACL might specify that only the user with a specific SID can access it.

Distinguished Name (DN)

A Distinguished Name (DN) **describes the full path to an object in AD** (such as **cn**=bjones**, ou**=IT**, ou**=Employees**, dc**=inlanefreight**, dc**=local). In this example, *the **user** bjones works in the **IT department** of the **company Inlanefreight**, and his account is created in an Organizational Unit (OU) that holds accounts for company **employees***. The Common Name (**CN**) bjones **is just one way the user object could be searched for or accessed within the domain.**

Relative Distinguished Name (RDN)

A Relative Distinguished Name (RDN) is **a single component of the Distinguished Name that identifies the object as unique from other objects at the current level in the naming hierarchy**. In our example, bjones (**CN**) is the **Relative Distinguished Name of the object**. **AD does not allow two objects with the same name under the same parent container**, but there can be two objects with the same RDNs (*in the example both DN have cn, dcs   but the schema is different*) that are still unique in the domain because they have different DNs. For example, the object **cn=bjones,dc=dev,dc=inlanefreight,dc=local** **would be recognized as different from** **cn=bjones,dc=inlanefreight,dc=local**.


Distinguished Name (DN) Relative Distinguished Name (RDN)
inlanefreight.local/Users/Sales/Managers/BJones
Inlanefreight.local
Users OU
Sales OU
Managers OU
BJones
DN must be unique in the directory.
RDN must be unique in an OU.

sAMAccountName

The sAMAccountName is **the user's logon name**. Here it would just be bjones . It must be a **unique value** and **20** or **fewer characters**.

- Represents the **pre-Windows 2000 logon name of the object** (refers to the **NetBIOS-style username** format that was used in older versions of Windows (prior to Windows 2000) for logging into systems and accessing resources within a domain.)
- Used primarily for compatibility with older systems and services.

Common Use:

- To log in to a domain or access resources (e.g., ***DOMAIN\sAMAccountName***).

userPrincipalName

The userPrincipalName attribute is another way to identify users in AD. This attribute consists of a prefix (**the user account name**) and a suffix (**the domain name**) in the format of ***bjones@inlanefreight.local***. This attribute is not mandatory.

- The domain portion does not have to match the actual AD domain. For instance:
  - Actual AD domain: ad.mydomain.local
  - UPN domain: mydomain.com

Attribute	**sAMAccountName**	**userPrincipalName (UPN)**
Purpose	Legacy logon name	Modern internet-style logon name
Format	**DOMAIN\sAMAccountName**	**username@domain**
Character Limit	20 characters	256 characters
Scope of Uniqueness	**Unique within the domain**	**Unique within the forest**
Preferred Usage	Legacy systems, older protocols	Modern systems, cloud services
Visibility	Often seen in on-prem environments	Commonly used in hybrid/cloud setups

Common Use Cases

1. **Logon Name:**
   - Users can log on with the UPN instead of DOMAIN\username.
     Example: Instead of logging in as MYDOMAIN\jdoe, the user can use jdoe@mydomain.com.
2. **SSO and Azure Integration:**
   - UPN is critical for integrating on-premises AD with cloud services like Microsoft 365 or Azure AD.
3. **Federation:**
   - Used for federation scenarios where SSO is implemented across different systems or organizations.
4. **Email Integration:**
   - UPN often matches the user’s primary email address for consistency.

FSMO Roles

In the early days of AD, **if you had multiple DCs in an environment**, they would fight over **which DC gets to make changes**, and sometimes changes would not be made properly. Microsoft then implemented *"last writer wins,"* which could introduce its own problems if the last change breaks things. They then **introduced a model in which a single *"master" DC* could apply changes to the domain while the others merely fulfilled authentication**

requests. This was a flawed design because if the master DC went down, no changes could be made to the environment until it was restored. **To resolve this single point of failure model, Microsoft separated the various responsibilities that a DC can have into Flexible Single Master Operation (FSMO) roles**. These **give Domain Controllers (DC) the ability to continue authenticating users and granting permissions without interruption (authorization and authentication)**. There are five FSMO roles:

FSMO Role	Scope	Key Responsibility
Schema Master	1 per Forest-wide	Manages AD schema changes
Domain Naming Master	1 per Forest-wide	Ensures unique domain names in the forest
RID Master	1 per Domain-wide	Allocates RIDs to ensure unique SIDs
PDC Emulator	1 per Domain-wide	Acts as the primary authentication source and time synchronizer
Infrastructure Master	1 per Domain-wide	Maintains cross-domain object references

All five roles **are assigned to the first DC in the forest root domain** in a new AD forest. **Each time a new domain is added to a forest**, **only the 3 domain-wide roles** ; RID Master, PDC Emulator, and Infrastructure Master roles **are assigned to the new domain**. FSMO roles are typically set when domain controllers are created, but sysadmins can transfer these roles if needed. **These roles help replication in AD to run smoothly** and **ensure that critical services are operating correctly**.

Best Practices:

1. **Placement**:
   - Keep **Schema Master** and **Domain Naming Master** roles on the same DC to simplify forest-wide changes.
   - Place the **Infrastructure Master** role on a non-Global Catalog (GC) server (unless all DCs in the domain are GCs).
   - The **PDC Emulator** should be on a reliable DC to ensure timely replication and time synchronization.
2. **Monitoring**:
   - Regularly monitor the availability and health of FSMO role holders.
   - Use tools like **dcdiag** to check FSMO role functionality.

Global Catalog

A global catalog (GC) is **a domain controller that stores copies of ALL objects in an Active Directory forest**. The GC stores a **full copy of all objects in the current domain** and **a partial copy of objects that belong to other domains in the forest**.

Difference : **Standard domain controllers hold a complete replica of objects belonging to its domain *but not those of different domains in the forest***. **The GC allows both users and applications to find information about any objects in ANY domain in the forest**. GC is a feature that is enabled on a domain controller and **performs the following functions**:

- **Authentication** (provided authorization for all groups that a user account belongs to, **which is included when an access token is generated**)
- **Object search** (making the directory structure within a forest transparent, **allowing a search to be carried out across all domains in a forest by providing just one attribute about an object**.)

Read-Only Domain Controller (RODC)

A Read-Only Domain Controller (RODC) is a type of domain controller (DC) introduced in **Windows Server 2008**, designed for scenarios where deploying a full, writable DC is not feasible or secure. It provides read-only access to Active Directory (AD) data and is ideal for remote or less-secure locations.

- Local admins can manage the RODC without affecting the overall domain.
- The AD database on an RODC is read-only, meaning it cannot make changes to the directory.
- Changes (like password updates or account modifications) must be sent to a writable DC, which then replicates those changes back to the RODC.
- One-way replication prevents any potential corruption or malicious changes from being propagated.

RODCs are designed for environments where physical security is limited (e.g., branch offices). Even if compromised, the damage is limited because:

- No sensitive credentials are stored locally (if caching is disabled).
- It cannot make changes to the AD database.

Example Use Case

Imagine a company with a headquarters in a secure location and a branch office in a remote area where physical security is poor. Deploying an RODC in the branch office ensures that:

- Local users can log in quickly using cached credentials. (Credentials can be explicitly allowed to be cached for specific accounts through a Password Replication Policy (PRP). This limits exposure to only those accounts, reducing the risk.)

- The branch office staff has limited administrative rights to the RODC without risking the security of the entire domain.
- If the RODC is compromised, the AD data it holds cannot be used to attack the rest of the domain.
  - Even if they access the read-only AD database, it does not allow them to make changes.
  - They cannot use the RODC to inject malicious data or commands into the AD domain.

Replication

Replication **happens in AD when AD objects are updated and transferred from one Domain Controller to another**. **Whenever a DC is added, connection objects are created to manage replication between them**. These connections are made by the **Knowledge Consistency Checker** (**KCC**) service, which is present on all DCs. Replication ensures that **changes are synchronized with all other DCs in a forest**, **helping to create a backup in case one domain controller fails.**

Service Principal Name (SPN)

A Service Principal Name (SPN) **uniquely identifies a service instance**. They are **used by Kerberos authentication** **to associate an instance of a service with a logon account**, **allowing** a **client application** to request the service to **authenticate an account** without needing to know the account name.

Group Policy Object (GPO)

Group Policy Objects (GPOs) are **virtual collections of policy settings**. Each GPO has a unique GUID. A GPO **can contain local file system settings** or **Active Directory settings**. GPO settings **can be applied to both *user*** and ***computer* objects**. They **can be applied to all users and computers within** the **domain** **or** defined more granularly at the **OU** level.

Access Control List (ACL)

An Access Control List (ACL) is the **ordered collection of Access Control Entries** (ACEs) **that apply to an object**.

Access Control Entries (ACEs)

Each Access Control Entry (ACE) in an ACL **identifies a trustee** (user account, group account, or logon session) and **lists the access rights** that are allowed, denied, or audited **for the given trustee.**

Discretionary Access Control List (DACL)

DACLs is a part of the Windows security model that **defines who can access a securable object** (like files, folders, or registry keys) and **what they are allowed to do with it**. It is essentially a list of rules that specify permissions for different users or groups; **it contains a list of ACEs**. When a process tries to access a securable object, the system checks the ACEs in the object's **DACL** to **determine whether or not to grant access.**
If an object does NOT have a DACL, **then the system will grant full access to everyone**, **but if the DACL has no ACE entries**, the system will deny all access attempts. ACEs in the DACL are checked in sequence until a match is found that allows the requested rights or until access is denied.

It is a list of **Access Control Entries (ACEs)**, where each ACE specifies:
- A **security principal** (a user, group, or service account).
- **The type of access (**allow or deny).
- Specific **permissions** (**read, write, execute**, etc.).

Example:
If you have a **file**, its DACL might look like this:
- **UserA**: **Allow Read**, **Write**
- **UserB**: **Deny Write**

System Access Control Lists (SACL)

is a component of the Windows security model that is **used for auditing access to securable objects**. Unlike a **DACL**, which grants or denies permissions, a SACL is **designed to log and monitor attempts to access an object** (whether successful or not, well to be more clear ACEs specify the types of access attempts that cause the system to generate a record in the security event log) .

Fully Qualified Domain Name (FQDN)

An FQDN **is the complete name for a specific computer** or **host**. It is written with the **hostname** and **domain name** in the format **[host name].[domain**

name].**[tld]**. This is **used to specify an object's location in the tree hierarchy of DNS**. The FQDN can be used to locate hosts in an Active Directory **without knowing the IP address**, much like when browsing to a website such as google.com instead of typing in the associated IP address. **An example would be** the host **DC01** in the domain **INLANEFREIGHT.LOCAL**. The FQDN here would be **DC01.INLANEFREIGHT.LOCAL**.

Tombstone

A tombstone is **a container object in AD that holds deleted AD objects**. **When an object is deleted from AD, the object remains for a set period of time known as the Tombstone Lifetime**, and the **isDeleted attribute** is set to **TRUE**. Once an object exceeds the Tombstone Lifetime, it will be entirely removed. Microsoft recommends a **tombstone lifetime of 180 days** to increase the usefulness of backups, but this value may differ across environments. Depending on the DC operating system version, this value will default to **60** or **180 days**. If an object is deleted in a domain that does not have an AD Recycle Bin, **it will become a tombstone object**. When this happens, **the object is stripped of most of its attributes** and **placed in the Deleted Objects container** for the **duration of the tombstone Lifetime**. **It can be recovered**, **but any attributes that were lost can no longer be recovered.**

AD Recycle Bin

The AD Recycle Bin was first introduced in Windows Server 2008 R2 to facilitate the recovery of deleted AD objects. This made it easier for sysadmins to restore objects, avoiding the need to restore from backups, restarting Active Directory Domain Services (AD DS), or rebooting a Domain Controller. When the AD Recycle Bin is enabled, **any deleted objects are preserved for a period of time**, **facilitating restoration if needed**. Sysadmins can set how long an object remains in a deleted, recoverable state. If this is not specified, the object will be restorable for a **default value of 60 days**. **The biggest advantage of using the AD Recycle Bin is that most of a deleted object's attributes are preserved**, which makes it far easier to fully restore a deleted object to its previous state.

Feature	**Tombstone**	**AD Recycle Bin**
State	Temporary marker indicating deleted object	Full recovery of deleted object, with all attributes preserved
Lifetime	Default is 180 days (tombstone lifetime)	Default is 60 days (configurable)
Object Information	Limited (some attributes are lost)	Full object attributes, including passwords
Recovery Process	Requires advanced tools (e.g., NTDSUtil)	Can be done via ADUC, PowerShell, or ADAC
Replication	Replicated across domain controllers	Replicated across domain controllers
Enabled by Default	Yes	No (must be manually enabled)
Recovery Ease	Difficult (requires command-line tools)	Easy and straightforward

SYSVOL

The SYSVOL **folder**, or **share**, **stores copies of public files in the domain** such as **system policies**, **Group Policy settings**, **logon/logoff scripts**, and **often contains other types of scripts that are executed to perform various tasks in the AD environment.** **The contents** of the SYSVOL folder **are replicated to all DCs within the environment using File Replication Services** (**FRS**). You can read more about the SYSVOL structure here.

AdminSDHolder

The AdminSDHolder is a **special object that ensures privileged accounts** (such as those in the **Domain Admins** or **Enterprise Admins** groups) **maintain consistent security settings**. It periodically applies security templates to these accounts **to prevent accidental modification of their permissions through inheritance or other means**. It acts as a ***container that holds the Security Descriptor applied to members of protected groups***. The **SDProp** (SD Propagator) **process runs on a schedule on the PDC Emulator Domain Controller**. When this process runs, **it checks members of protected groups to ensure that the correct ACL is applied to them**. **It runs every hour by default**.
For example, suppose an attacker is able to create a malicious ACL entry to grant a user certain rights over a member of the Domain Admins group. In that case, unless they modify other settings in AD, these rights will be removed (and they will lose any persistence they were hoping to achieve) when the SDProp process runs on the set interval.

dsHeuristics

The dsHeuristics **attribute** is a **string value** set on the **Directory Service** object **used to define multiple forest-wide configuration settings**. One of these settings is to **exclude built-in groups from the Protected Groups** list. Groups in this list are protected from modification **via** the AdminSDHolder object. If a group is excluded via the dsHeuristics attribute, then any changes that affect it will not be reverted when the SDProp process runs.
Common Use Cases for dsHeuristics

1. **Windows 2000 and 2003 Interoperability**:
   - The **dsHeuristics** attribute is commonly used when dealing with domain controllers running older versions of Windows Server (such as Windows 2000 or 2003) in environments with newer versions (such as Windows Server 2008 or later). It helps ensure that newer domain controllers can interact with older ones and make adjustments to replication settings that are required for interoperability.
2. **Behavior with Forest and Domain Functional Levels**:
   - The attribute can be set to accommodate changes that occur when you raise the **forest** or **domain functional levels** in Active Directory. Certain features might behave differently depending on these levels, and **dsHeuristics** allows you to fine-tune those settings.
3. **Domain Controller Promotion and Demotion**:
   - In environments where domain controllers are being promoted or demoted, **dsHeuristics** can influence the overall behavior of the domain controller during these operations, ensuring proper replication and consistency across all domain controllers.

adminCount

The adminCount attribute **determines whether or not the SDProp process protects a user**. It identifies objects (such as users or groups) that are considered "administrators" or are members of high-privilege groups like **Domain Admins**, **Enterprise Admins**, or **Administrators**. The **adminCount** attribute is set to **1** for these privileged accounts and **0** for non-privileged accounts.

Attackers will often look for accounts with the adminCount attribute set to 1 to target in an internal environment. These are often privileged accounts and may lead to further access or full domain compromise.

Active Directory Users and Computers (ADUC)

ADUC is **a GUI console** commonly **used for managing** users, groups, computers, and contacts **in AD**. **Changes made in ADUC can be done via PowerShell as well.**

How to Access ADUC

ADUC is part of the Remote Server Administration Tools (RSAT) package, which must be installed on client machines (typically Windows 10 or later) if they are not domain controllers.

i. Windows Server:
   - If you're using a Windows Server machine (especially one functioning as a domain controller), ADUC can be accessed by opening Server Manager, then going to Tools > Active Directory Users and Computers.

ii. Windows 10/11 (with RSAT):
   - On Windows 10 or later, you can install RSAT via the Optional Features settings. After installation, ADUC will appear under Administrative Tools or can be accessed by searching for "Active Directory Users and Computers" in the Start menu.

iii. Search in Run or Start Menu:
   - On a server, you can also type dsa.msc into the Run dialog (Win + R) or search for it in the Start menu to open ADUC directly.

Common Tasks :

- **Creating a User**:
  - Right-click on an organizational unit (OU) or container where you want to create the user.
  - Select **New > User**, and follow the prompts to enter the user's first name, last name, username, and other required information.
- **Creating a Group**:
  - Right-click the OU or container where you want to create the group.
  - Select **New > Group**, choose the group type (Security or Distribution), and configure group memberships.
- **Resetting a Password**:
  - Right-click the user account whose password you want to reset.
  - Select **Reset Password**, enter the new password, and confirm.
- **Enabling or Disabling Accounts**:
  - Right-click the user or computer account and select **Enable Account** or **Disable Account** depending on the status you want to set.
- **Moving Objects**:
  - To move users, computers, or groups between OUs, drag the object to a new location, or use the **Move** option from the object's context menu.
- **Viewing Object Properties**:
  - Right-click an object (user, group, computer, etc.) and select **Properties**. From here, you can modify the object's attributes (such as email, office, description, etc.).
  - Advanced features, such as the **Attribute Editor** tab, are available when you enable **Advanced Features** from the **View** menu.

ADSI Edit

ADSI Edit is **a GUI tool used to manage objects in AD.** It **provides access to far more than is available in** ADUC and can be used to set or delete any attribute available on an object, add, remove, and move objects as well. It is a powerful tool that **allows a user to access AD at a much deeper level**. Great care should be taken when using this tool, as changes here could cause major problems in AD.

How to Access ADSI Edit

ADSI Edit is a Microsoft Management Console (MMC) snap-in that is included with the **Remote Server Administration Tools (RSAT)**. Here's how you can access ADSI Edit:

1. **On a Domain Controller**:
   - If you're on a domain controller or a server that has **RSAT** installed, you can open ADSI Edit by typing adsiedit.msc into the **Run** dialog (Win + R) or search for **ADSI Edit** in the Start menu.
2. **On a Non-Domain Controller**:
   - If you're using a client machine like **Windows 10/11**, you first need to install the **RSAT** tools. After installation, you can open ADSI Edit by typing adsiedit.msc into the **Run** dialog or the Start menu.
3. **Launch via MMC**:
   - You can also add **ADSI Edit** as a snap-in through the **Microsoft Management Console (MMC)**:
     - Open **MMC** (type mmc in the Run dialog).
     - From the **File** menu, select **Add/Remove Snap-in**.
     - Choose **ADSI Edit** and click **Add**, then click **OK**.

sIDHistory

This attribute ***holds any SIDs that an object was assigned previously***. It is usually **used in migrations so a user can maintain the same level of access when migrated from one domain to another**. This attribute **can potentially be abused if set insecurely**, allowing an attacker to gain prior elevated access that an account had before a migration **if SID Filtering** (or removing SIDs from another domain from a user's access token that could be used for elevated access) **is not enabled**.

NTDS.DIT

The NTDS.DIT file can be considered the heart of Active Directory. It is **stored on a Domain Controller** at **C:\Windows\NTDS\** and is **a database that stores AD data** such as **information** about **user** and **group objects**, **group membership**, and, most important to attackers and penetration testers, **the password hashes for all users in the domain**. Once full domain compromise is reached, an attacker can retrieve this file, extract the hashes, and either use them to perform a pass-the-hash attack or crack them offline using a tool such as Hashcat to access additional resources in the domain. **If the setting Store password with reversible encryption is enabled,** then the NTDS.DIT will also store the cleartext passwords for all users created or who changed their password after this policy was set. While rare, some organizations may enable this setting if they use applications or protocols that need to use a user's existing password (and not Kerberos) for authentication.

MSBROWSE

MSBROWSE is **a Microsoft networking protocol** that was **used in early versions of Windows-based local area networks** (LANs) to **provide browsing services**. It was used to **maintain a list of resources**, such as **shared printers** and **files**, **that were available on the network**, and to **allow users to easily browse and access these resources**.

In older version of Windows we could use `nbtstat -A ip-address` **to search for the Master Browser**. If we see MSBROWSE it means that's the Master Browser. Aditionally we could use `nltest` utility to **query a Windows Master Browser for the names of the Domain Controllers.**

Today, MSBROWSE is largely obsolete and is no longer in widespread use. **Modern Windows-based LANs use the Server Message Block (SMB) protocol for file and printer sharing**, **and** the Common Internet File System **(CIFS) protocol for browsing services.**

Active Directory Objects

We will often see the term "objects" when referring to AD. What is an object?

An object can be defined as **ANY resource present within an Active Directory environment** such as OUs, printers, users, domain controllers.

Active Directory Objects
Domain
Computer
OU
Groups
User
Printers

Users

These are the users within the organization's AD environment. **Users are considered leaf objects**, which means that **they cannot contain any other objects within them**. **Another example** of a leaf object **is a mailbox in Microsoft Exchange**.
A user object **is considered a security principal** and **has a security identifier (SID) and a global unique identifier (GUID)**. User objects **have many possible attributes**, such as their display name, last login time, date of last password change, email address, account description, manager, address, and more. Depending on how a particular Active Directory environment is set up, **there can be over 800 possible user attributes** when accounting for ALL possible attributes as detailed here. This example goes far beyond what is typically populated for a standard user in most environments but shows Active Directory's sheer size and complexity. They are a crucial target for attackers since gaining access to even a low privileged user can grant access to many objects and resources and allow for detailed enumeration of the entire domain (or forest).

Contacts

A contact object **is usually used to represent an external user** and **contains informational attributes such** as first name, last name, email address, telephone number, etc. They are **leaf objects** and are **NOT security principals** (securable objects), so **they don't have a SID**, **only a GUID**. An example would be a contact card for a third-party vendor or a customer.

Printers

A printer object **points to a printer accessible within the AD network. Like a contact**, a printer is **a leaf object** and **NOT a security principal**, so **it only has a GUID**. Printers have attributes such as the printer's name, driver information, port number, etc.

Computers

A computer object is **any computer joined to the AD network** (**workstation** or **server**). Computers **are leaf objects** because they do not contain other objects. However, they are **considered security principals** and **have a SID** and **a GUID**. Like users, they are prime targets for attackers since full administrative access to a computer (as the all-powerful **NT AUTHORITY\SYSTEM** account) **grants similar rights to a standard domain user** and can be used to perform the majority of the enumeration tasks that a user account can (save for a few exceptions across domain trusts.)
While NT AUTHORITY\SYSTEM is all-powerful locally, it is not inherently a domain user. Domain-related actions require domain credentials or privileges. However, **SYSTEM can interact with the domain using the computer's machine account** (e.g., *DOMAIN\ComputerName$*), **which has certain rights within the domain but is not equivalent to a domain user.**

Shared Folders

A shared folder object **points to a shared folder on the specific computer where the folder resides**. Shared folders can have stringent access control applied to them and can be either accessible **to everyone** (even those without a valid AD account), open **to only authenticated users** (which means anyone with **even the lowest privileged** user account **OR** a computer account (NT AUTHORITY\SYSTEM) could access it), or **be locked down to only allow certain users/groups access**. Anyone not explicitly allowed access will be denied from listing or reading its contents. Shared folders **are NOT security principals** and **only have a GUID**. A shared folder's attributes can include the name, location on the system, security access rights.

Groups

A group is considered **a container object** because **it can contain** other objects, including **users**, **computers**, and ***even other groups***. A group IS regarded as **a security principal** and **has a SID** and **a GUID**. In AD, groups are **a way to manage user permissions** and **access to other securable objects** (both users and computers). Let's say we want to give 20 help desk users access to the Remote Management Users group on a jump host. Instead of adding the users one by one, we could add the group, and the users would inherit the intended permissions via their membership in the group. In Active Directory, we commonly see what are called "**nested groups**" (a group added as a member of another group), **which can lead to a user(s) obtaining unintended rights**. Nested group membership is something we see and often leverage during penetration tests. The tool BloodHound helps to discover attack paths within a network and illustrate them in a graphical interface. It is excellent for auditing group membership and uncovering/seeing the sometimes unintended impacts of nested group membership. Groups in AD can have many attributes, the most common being the name, description, membership, and other groups that the group belongs to. Many other attributes can be set, which we will discuss more in-depth later in this module.

Organizational Units (OUs)

An organizational unit, or OU from here on out, is **a container that systems administrators can use to store similar objects for ease of administration**. OUs are often used for administrative delegation of tasks without granting a user account full administrative rights. For example, we may have a top-level OU called Employees and then child OUs under it for the various departments such as Marketing, HR, Finance, Help Desk, etc. If an account were given the right to reset passwords over the top-level OU, this user would have the right to reset passwords for all users in the company. However, if the OU structure were such **that specific departments were child** OUs **of the Help Desk OU**, then **any user placed in the Help Desk OU would have this right delegated to them if granted**. Other tasks that may be delegated at the OU level include creating/deleting users, modifying group membership, managing Group Policy links, and performing password resets. OUs are very useful for managing Group Policy (which we will study later in this module) settings across a subset of users and groups within a domain. For example, we may want to set a specific password policy for privileged service accounts so these accounts could be placed in a particular OU and then have a Group Policy object assigned to it, which would enforce this password policy on all accounts placed inside of it. A few OU attributes include its name, members, security settings, and more.

Feature	**Groups**	**Organizational Units (OUs)**
Purpose	Manage permissions and access.	Organize objects in AD.
Contains	Users, computers, other groups.	Users, groups, computers, OUs.
Used For	Controlling resource access.	Structuring AD and applying GPOs.
Hierarchy	No hierarchy (flat structure).	Hierarchical (can have sub-OUs).
Policies	No direct policies.	GPOs can be applied.

Domain

A domain **is the structure of an AD network**. Domains **contain objects** such as **users** and **computers**, which are **organized into container objects**: **groups** and **OUs**. **Every domain has its own separate database and sets of policies** that can be applied to any and all objects within the domain. **Some policies are set by default** (and can be tweaked), such as the domain password policy. In contrast, others are created and applied based on the organization's need, *such as blocking access to cmd.exe for all non-administrative users* or *mapping shared drives at log in*.

Domain Controllers

Domain Controllers are essentially **the brains of an AD network**. They **handle authentication requests**, **verify users on the network**, and **control who can access the various resources in the domain**. All access requests are validated via the domain controller and privileged access requests are based on predetermined roles assigned to users. It also **enforces security policies** and **stores information about every other object in the domain**.

Sites

A site in AD is a **set of computers across one or more subnets connected using high-speed links**. They are **used to make replication across domain controllers run efficiently.**

Built-in

In AD, built-in is **a container that holds default groups in an AD domain**. They are predefined when an AD domain is created.

Foreign Security Principals

A foreign security principal (FSP) is **an object created in AD to represent a security principal that belongs to a trusted external forest**. They are **created when an objec**t such as a user, group, or computer **from an external** (outside of the current) **forest is added to a group in the current domain**. They are created automatically after adding a security principal to a group. Every foreign security principal **is a placeholder object** that **holds the SID of the foreign object** (an object that belongs to another forest.) **Windows uses this SID to resolve the object's name** via the trust relationship. FSPs are created in a specific container named **ForeignSecurityPrincipals** with a distinguished name like cn=**ForeignSecurityPrincipals**,dc=inlanefreight,dc=local.

Active Directory Functionality

As mentioned before, there are five Flexible Single Master Operation (FSMO) roles. These roles can be defined as follows:

Roles	Description
Schema Master	This role manages the read/write copy of the AD schema, which defines all attributes that can apply to an object in AD. Initially assigned to the first domain controller in the forest root domain.
Domain Naming Master	Manages domain names and **ensures that two domains of the same name are not created in the same forest**. Assigned to the first domain controller in the forest root domain during forest creation.
Relative ID (RID) Master	The RID Master assigns blocks of RIDs to other DCs within the domain that can be used for creating security principals like users, groups, and computers.. The RID Master helps **ensure that multiple objects are not assigned the same SID**. **Domain object SIDs** are the **domain SID** combined with the **RID number assigned to the object** to make the unique SID. Initially assigned to the first domain controller in the domain.
PDC Emulator	The host with this role would be **the authoritative DC in the domain** and **respond to authentication requests**, **password changes**, and **manage Group Policy Objects** (GPOs). The PDC Emulator also **maintains time within the domain**. Initially assigned to the first domain controller in the domain.
Infrastructure Master	This role **translates GUIDs, SIDs, and DNs between domains**. This role is **used in organizations with multiple domains in a single forest**. The Infrastructure Master helps them to communicate. If this role is not functioning properly, Access Control Lists (ACLs) will show SIDs instead of fully resolved names. Initially assigned to the first domain controller in the domain.

Depending on the organization, these roles may be assigned to specific DCs or as defaults each time a new DC is added. Issues with FSMO roles will lead to authentication and authorization difficulties within a domain.

Domain and Forest Functional Levels

Microsoft **introduced functional levels** to determine the various features and capabilities available in Active Directory Domain Services (AD DS) at the domain and forest level. They are also **used to specify which Windows Server operating systems can run a Domain Controller** in a domain or forest. This and this article describe both the domain and forest functional levels from Windows 2000 native to Windows Server 2012 R2;

- if you are sure that you will never add domain controllers that run Windows Server 2003 to the domain or forest, select the Windows Server 2008 functional level during the deployment process.
- When you deploy a new forest, you are prompted to set the forest functional level and then set the domain functional level :
  - You **cannot** set the **domain** functional **level** to a value that is **lower** than the **forest** functional **level**
  - You **can** set the **domain** functional level to a value that is **higher** than the **forest** functional **level**.

Below is a quick overview of the differences in domain functional levels from Windows 2000 native up to Windows Server 2016, aside from all default Active Directory Directory Services features from the level just below it (or just the default AD DS features in the case of Windows 2000 native.)

Domain Functional Level	Features Available	Supported Domain Controller Operating Systems
Windows 2000 native	Universal groups for distribution and security groups, group nesting, group conversion (between security and distribution and security groups), SID history , **use Kerberos as the primary authentication protocol** .	Windows Server 2008 R2, Windows Server 2008, Windows Server 2003, Windows 2000 Means up to WS 2008 can operate with WS 2000.
Windows Server 2003	Netdom.exe domain management tool, lastLogonTimestamp attribute introduced, well-known users and computers containers, constrained delegation, selective authentication.	Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, Windows Server 2008, Windows Server 2003
Windows Server 2008	Distributed File System **(DFS) replication support**, **Advanced Encryption Standard** (AES 128 and AES 256) **support for the Kerberos protocol,** Fine-grained password policies	Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2,

		Windows Server 2008
Windows Server 2008 R2	Authentication mechanism assurance, Managed Service Accounts	Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2
Windows Server 2012	**KDC support for claims**, compound authentication, and **Kerberos armoring**	Windows Server 2012 R2, Windows Server 2012
Windows Server 2012 R2	Extra protections for members of the Protected Users group, Authentication Policies, Authentication Policy Silos	Windows Server 2012 R2
Windows Server 2016	Smart card required for interactive logon **new Kerberos features** and new credential protection features *Smart cards that require personal identification numbers (PINs) provide two-factor authentication: the user who attempts to sign in must possess the smart card and know its PIN. A malicious user who captures the authentication traffic between the user's device and the domain controller will find it difficult to decrypt the traffic: even if they do, the next time the user signs in to the network, a new session key will be generated for encrypting traffic between the user and the domain controller.*	Windows Server 2019 and Windows Server 2016

A new functional level was not added with the release of Windows Server 2019. However, **Windows Server 2008 functional level is the minimum requirement for adding Server 2019 Domain Controllers** to an environment. Also, the target domain has to use DFS-R (DFS Replication : Applies To: Windows Server 2022, Windows Server 2019, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, Windows Server 2008) **for SYSVOL replication.**

Forest *functional levels have introduced a few key capabilities over the years:*

Version	**Capabilities**
Windows Server 2003	saw the introduction of the forest trust, domain renaming, read-only domain controllers (RODC), and more.
Windows Server 2008	All new domains added to the forest default to the Server 2008 domain functional level. No additional new features.
Windows Server 2008 R2	Active Directory Recycle Bin provides the ability to restore deleted objects when AD DS is running.
Windows Server 2012	All new domains added to the forest default to the Server 2012 domain functional level. No additional new features.
Windows Server 2012 R2	All new domains added to the forest default to the Server 2012 R2 domain functional level. No additional new features.
Windows Server 2016	Privileged access management (PAM) using Microsoft Identity Manager (MIM).

Trusts

A trust is used to establish **forest-forest** or **domain-domain** **authentication**, allowing users to access resources in (or administer) another domain outside of the domain their account resides in. A trust **creates a link between the authentication systems of two domains.**

Here's an example of a hierarchy in an Active Directory forest with a new tree root domain:

Forest Root Domain: example.com

- This is the first domain created during the setup of the AD forest.
- **Domain controllers in this domain manage the overall forest structure and schema.**

```
example.com
  |— Users
  |— Computers
  └── Domain Controllers
```

Adding a New Tree Root Domain

This domain is part of the same forest but has a completely separate DNS namespace (newtree.local) from the forest root domain (example.com).

A child domain under the new tree root domain, used for managing a specific department in the new namespace.

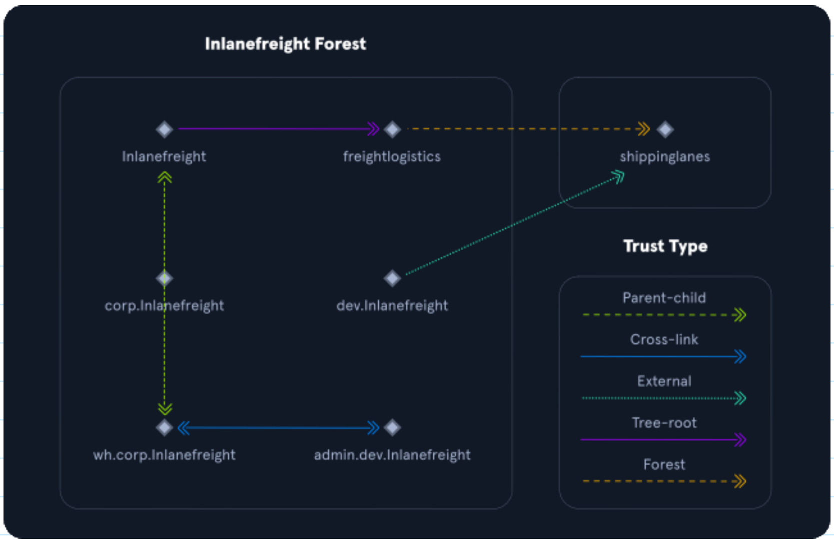
```
Forest: example.com
  |— example.com
  |    |— sales.example.com
  |    └── Domain Controllers
  |
  └── newtree.local
       |— marketing.newtree.local
       |    |— Users
       |    |— Computers
       |    └── Domain Controllers
```

└── Users (within newtree.local)

The forest root domain (example.com), the child domain (sales.example.com), and the new tree root domain (newtree.local) automatically share **two-way transitive trusts** **(** because they belong to the same Active Directory forest) , allowing users in one domain to access resources in another (depending on permissions).

There are several trust types.

Trust Type	**Description**
Parent-child	Domains within the same forest. The child domain has a two-way transitive trust with the parent domain. • ***Example.com*** **↔** ***sales.example.com***
Cross-link	a trust between child domains to speed up authentication. • **sales.example.com ↔ marketing.newtree.local**
External	A non-transitive trust <u>between two separate domains in separate forests</u> which are not already joined by a forest trust. **This type of trust utilizes SID filtering.** • **example.com ↔ legacydomain.local**
Tree-root	a two-way transitive trust between a **forest root domain** and **a new tree root domain**. They are created by design when you set up a new tree root domain within a forest. • ***Example.com*** *and* ***newtree.local***
Forest	a transitive trust between two forest root domains. • ***example.com (Forest A) ↔ anotherforest.com (Forest B)***


Inlanefreight Forest
Inlanefreight
freightlogistics
shippinglanes
corp.Inlanefreight
dev.Inlanefreight
wh.corp.Inlanefreight
admin.dev.Inlanefreight
Trust Type
Parent-child
Cross-link
External
Tree-root
Forest