AWS Directory Service

Administration Guide

Version 1.0

AWS Directory Service: Administration Guide

Copyright © 2017 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is AWS Directory Service?

AWS Directory Service provides multiple ways to use Amazon Cloud Directory and Microsoft Active Directory (AD) with other AWS services. Directories store information about users, groups, and devices, and administrators use them to manage access to information and resources. AWS Directory Service provides multiple directory choices for customers who want to use existing Microsoft AD or Lightweight Directory Access Protocol (LDAP)–aware applications in the cloud. It also offers those same choices to developers who need a directory to manage users, groups, devices, and access.

Which to Choose

You can choose directory services with the features and scalability that best meets your needs. Use the following table to help you determine which AWS Directory Service directory option works best for your organization.

What do you need to do?	Recommended AWS Directory Service options
I need Active Directory or LDAP for my applications in the cloud	Select AWS Directory Service for Microsoft Active Directory (p. 1) (Enterprise Edition) if you need an actual Microsoft Active Directory in the AWS Cloud that supports Active Directory–aware workloads, or AWS applications and services such as Amazon WorkSpaces and Amazon QuickSight, or you need LDAP support for Linux applications. Use AD Connector (p. 2) if you only need to allow your on-premises users to log in to AWS applications and services with their Active Directory credentials. You can also use AD Connector to join Amazon EC2 instances to your existing Active Directory domain. Use Simple AD (p. 3) if you need a low-scale, low-cost directory with basic Active Directory compatibility that supports Samba 4–compatible applications, or you need LDAP compatibility for LDAP-aware applications.
I develop cloud applications that manage hierarchical data with complex relationships	Use Amazon Cloud Directory (p. 3) if you need a cloud-scale directory to share and control access to hierarchical data between your applications.
I develop SaaS applications	Use Amazon Cognito (p. 4) if you develop high-scale SaaS applications and need a scalable directory to manage and authenticate your subscribers and that works with social media identities.

AWS Directory Service Options

AWS Directory Service includes several directory types to choose from.

AWS Directory Service for Microsoft Active Directory

Also known as AWS Microsoft AD, AWS Directory Service for Microsoft Active Directory is powered by an actual Microsoft Windows Server Active Directory (AD) in the AWS Cloud. It includes key features, such as schema extensions, with which you can migrate a broad range of Active Directory–aware applications to the AWS Cloud. AWS Microsoft AD works with Microsoft SharePoint, Microsoft SQL Server Always On Availability Groups, and many .NET applications. It includes security features, such as fine-grained password policy management, and LDAP encryption through Secure Socket Layer (SSL)/Transport Layer Security (TLS). It is also approved for applications in the AWS Cloud that are subject to U.S. Health Insurance Portability and Accountability Act (HIPAA) or Payment Card Industry Data Security Standard (PCI DSS) compliance. AWS provides monitoring, daily snapshots, and recovery as part of the service.

You can use AWS Microsoft AD as a standalone Active Directory to administer users, groups, and computers in the AWS Cloud. With AWS Microsoft AD, you can also set up trust relationships to extend authentication from your existing on-premises Active Directory into the cloud. When used with a trust, your on-premises users can access Windows workloads running on Amazon EC2 with the same single sign-on (SSO) experience as when they access workloads in your on-premises network. When used as a standalone directory, your users can access third-party cloud applications such as Microsoft Office 365, through federation.

AWS Microsoft AD supports AWS applications and services including Amazon WorkSpaces, Amazon WorkDocs, Amazon QuickSight, Amazon Chime, Amazon Connect, and Amazon Relational Database Service for Microsoft SQL Server (RDS for SQL Server). You can also use your Active Directory credentials to authenticate to the AWS Management Console without having to set up a SAML authentication infrastructure. Because Active Directory is an LDAP directory, you can also use AWS Microsoft AD for Linux Secure Shell (SSH) authentication and for other LDAP-enabled applications.

AWS Microsoft AD is also scalable. You can increase the performance and redundancy of your directory by adding domain controllers. This can help improve application performance by enabling directory clients to load balance their request across a larger number of domain controllers. For more information, see Deploy Additional Domain Controllers (p. 114).

When deployed with AWS applications you use your existing RADIUS-based multi-factor authentication (MFA) infrastructure to provide an additional layer of security. For more information, see Multi-Factor Authentication (p. 109).

AWS Microsoft AD is available in two editions: Standard and Enterprise.

- **Standard Edition:** AWS Microsoft AD (Standard Edition) is optimized to be a primary directory for small and midsize businesses with up to 5,000 employees. It provides you enough storage capacity to support up to 30,000* directory objects, such as users, groups, and computers.
- **Enterprise Edition:** AWS Microsoft AD (Enterprise Edition) is designed to support enterprise organizations with up to 500,000* directory objects.

* Upper limits are approximations. Your directory may support more or less directory objects depending on the size of your objects and the behavior and performance needs of your applications.

When to use

Microsoft AD is your best choice if you need actual Active Directory features to support AWS applications or Windows workloads, including Amazon Relational Database Service for Microsoft SQL Server. It's also best if you want a standalone AD in the AWS Cloud that supports Office 365 or you need an LDAP directory to support your Linux applications. For more information, see Microsoft Active Directory (p. 55).

AD Connector

AD Connector is a proxy service that provides an easy way to connect compatible AWS applications, such as Amazon WorkSpaces and Amazon QuickSight, and Amazon EC2 for Windows Server instances, to your existing on-premises Microsoft Active Directory. With AD Connector , you can simply add one service account to your Active Directory. AD Connector also eliminates the need of directory synchronization or the cost and complexity of hosting a federation infrastructure.

When you add users to AWS applications such as Amazon QuickSight, AD Connector reads your existing Active Directory to create lists of users and groups to select from. When users log in to the AWS applications, AD Connector forwards sign-in requests to your on-premises Active Directory domain controllers for authentication. AD Connector works with many AWS applications and services including as Amazon WorkSpaces, Amazon WorkDocs, Amazon QuickSight, Amazon Chime, Amazon Connect, and Amazon WorkMail. You can also join your EC2 Windows instances to your on-premises Active Directory domain through AD Connector using seamless domain join. AD Connector also allows your users to access the AWS Management Console and manage AWS resources by logging in with their existing Active Directory credentials. AD Connector is not compatible with RDS SQL Server.

You can also use AD Connector to enable multi-factor authentication for your AWS application users by connecting it to your existing RADIUS-based MFA infrastructure. This provides an additional layer of security when users access AWS applications. For more information, see Enable Multi-Factor Authentication for AD Connector (p. 130).

With AD Connector, you continue to manage your Active Directory as you do now. For example, you add new users and groups and update passwords using standard Active Directory administration tools in your on-premises Active Directory. This helps you consistently enforce your security policies, such as password expiration, password history, and account lockouts, whether users are accessing resources on premises or in the AWS Cloud.

When to use

AD Connector is your best choice when you want to use your existing on-premises directory with compatible AWS services. For more information, see Active Directory Connector (p. 118).

Simple AD

Simple AD is a Microsoft Active Directory–compatible directory from AWS Directory Service that is powered by Samba 4. Simple AD supports basic Active Directory features such as user accounts, group memberships, joining a Linux domain or Windows based EC2 instances, Kerberos-based SSO, and group policies. AWS provides monitoring, daily snap-shots, and recovery as part of the service.

Simple AD is a standalone directory in the cloud, where you create and manage user identities and manage access to applications. You can use many familiar Active Directory–aware applications and tools that require basic Active Directory features. Simple AD is compatible with following AWS applications: Amazon WorkSpaces, Amazon WorkDocs, Amazon WorkMail, and Amazon QuickSight. You can also sign in to the AWS Management Console with Simple AD user accounts and to manage AWS resources.

Simple AD does not support trust relationships, DNS dynamic update, schema extensions, multi-factor authentication, communication over LDAPS, PowerShell AD cmdlets, or FSMO role transfer. Simple AD is not compatible with RDS SQL Server. Customers who require the features of an actual Microsoft Active Directory, or who envision using their directory with RDS SQL Server should use AWS Microsoft AD instead. Please verify your required applications are fully compatible with Samba 4 before using Simple AD. For more information, see https://www.samba.org.

When to use

You can use Simple AD as a standalone directory in the cloud to support Windows workloads that need basic AD features, compatible AWS applications, or to support Linux workloads that need LDAP service. For more information, see Simple Active Directory (p. 132).

Amazon Cloud Directory

Amazon Cloud Directory is a cloud-native directory that can store hundreds of millions of application-specific objects with multiple relationships and schemas. Use Amazon Cloud Directory if you need a highly scalable directory store for your application's hierarchical data.

When to use

Amazon Cloud Directory is a great choice when you need to build application directories such as device registries, catalogs, social networks, organization structures, and network topologies. For more information, see Amazon Cloud Directory (p. 11).

Amazon Cognito

Amazon Cognito is a user directory that adds sign-up and sign-in to your mobile app or web application using Amazon Cognito User Pools.

When to use

You can also use Amazon Cognito when you need to create custom registration fields and store that metadata in your user directory. This fully managed service scales to support hundreds of millions of users. For more information, see Creating and Managing User Pools.

Working with Amazon EC2

A basic understanding of Amazon EC2 is essential to using AWS Directory Service. We recommend that you begin by reading the following topics:

- What is Amazon EC2? in the *Amazon EC2 User Guide for Windows Instances*.
- Launching EC2 Instances in the *Amazon EC2 User Guide for Windows Instances*.
- Security Groups in the *Amazon EC2 User Guide for Windows Instances*.
- What is Amazon VPC? in the *Amazon VPC User Guide*.
- Adding a Hardware Virtual Private Gateway to Your VPC in the *Amazon VPC User Guide*.

Setting Up

To work with AWS Directory Service, you need to meet the prerequisites for AWS Directory Service for Microsoft Active Directory, AD Connector, or Simple AD. For more information, see Microsoft AD Prerequisites (p. 56), AD Connector Prerequisites (p. 119), or Simple AD Prerequisites (p. 133).

If you haven't already done so, you'll also need to create an AWS account and use the AWS Identity and Access Management service to control access.

Topics

Sign Up for AWS

Your AWS account gives you access to all services, but you are charged only for the resources that you use.

If you do not have an AWS account, use the following procedure to create one.

To sign up for AWS

1. Open https://aws.amazon.com/ and choose **Create an AWS Account**.
2. Follow the online instructions.

Your root account credentials identify you to services in AWS and grant you unlimited use of your AWS resources, such as your WorkSpaces. To allow other users to manage AWS Directory Service resources without sharing your security credentials, use AWS Identity and Access Management (IAM). We recommend that everyone work as an IAM user, even the account owner. You should create an IAM user for yourself, give that IAM user administrative privileges, and use it for all your work.

Create an IAM User

The AWS Management Console requires your username and password so that the service can determine whether you have permission to access its resources. However, we recommend that you avoid accessing AWS using the credentials for your root AWS account; instead, we recommend that you use AWS Identity and Access Management (IAM) to create an IAM user and add the IAM user to an IAM group with administrative permissions. This grants the IAM user administrative permissions. You then access the AWS Management Console using the credentials for the IAM user.

If you signed up for AWS but have not created an IAM user for yourself, you can create one using the IAM console.

To create an IAM user for yourself and add the user to an Administrators group

1. Use your AWS account email address and password to sign in to the AWS Management Console as the *AWS account root user*.
2. In the navigation pane of the console, choose **Users**, and then choose **Add user**.
3. For **User name**, type `Administrator`.

4. Select the check box next to **AWS Management Console access**, select **Custom password**, and then type the new user's password in the text box. You can optionally select **Require password reset** to force the user to select a new password the next time the user signs in.
5. Choose **Next: Permissions**.
6. On the **Set permissions for user** page, choose **Add user to group**.
7. Choose **Create group**.
8. In the **Create group** dialog box, type **`Administrators`**.
9. For **Filter**, choose **Job function**.
10. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.
11. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
12. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users, and to give your users access to your AWS account resources. To learn about using policies to restrict users' permissions to specific AWS resources, go to Access Management and Example Policies.

To sign in as this new IAM user, sign out of the AWS Management Console, then use the following URL, where *your_aws_account_id* is your AWS account number without the hyphens (for example, if your AWS account number is `1234-5678-9012`, your AWS account ID is `123456789012`):

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

Enter the IAM user name and password that you just created. When you're signed in, the navigation bar displays "*your_user_name @ your_aws_account_id*".

If you don't want the URL for your sign-in page to contain your AWS account ID, you can create an account alias. From the IAM dashboard, click **Customize** and enter an alias, such as your company name. To sign in after you create an account alias, use the following URL:

```
https://your_account_alias.signin.aws.amazon.com/console/
```

For more information about using IAM policies to control access to your AWS Directory Service resources, see Identity-Based Policies (IAM Policies) (p. 178).

AWS Directory Service Best Practices

Here are some suggestions and guidelines you should consider to avoid problems and get the most out of AWS Directory Service.

Setting Up: Prerequisites

Consider these guidelines before creating your directory.

Choose the Right Directory Type

AWS Directory Service provides multiple ways to use Microsoft Active Directory with other AWS services. You can choose the directory service with the features you need at a cost that fits your budget:

- **AWS Directory Service for Microsoft Active Directory** is a feature-rich managed Microsoft Active Directory hosted on the AWS cloud. Microsoft AD is your best choice if you have more than 5,000 users and need a trust relationship set up between an AWS hosted directory and your on-premises directories.
- **AD Connector** simply connects your existing on-premises Active Directory to AWS. AD Connector is your best choice when you want to use your existing on-premises directory with AWS services.
- **Simple AD** is an inexpensive Active Directory–compatible service with the common directory features. In most cases, Simple AD is the least expensive option and your best choice if you have 5,000 or fewer users and don't need the more advanced Microsoft Active Directory features.

For a more detailed comparison of AWS Directory Service options, see Which to Choose (p. 1).

Ensure Your VPCs and Instances are Configured Correctly

In order to connect to, manage, and use your directories, you must properly configure the VPCs that the directories are associated with. See either Microsoft AD Microsoft AD Prerequisites (p. 56), AD Connector AD Connector Prerequisites (p. 119), or Simple AD Simple AD Prerequisites (p. 133) for information about the VPC security and networking requirements.

If you are adding an instance to your domain, ensure that you have connectivity and remote access to your instance as described in Add an Instance to Your Directory (Simple AD and Microsoft AD) (p. 155).

Configure On-premises Sites and Subnets Correctly When Using AD Connector

If your on-premises network has Active Directory sites defined, you must make sure the subnets in the VPC where your AD Connector resides are defined in an Active Directory site, and that no conflicts exist between the subnets in your VPC and the subnets in your other sites.

To discover domain controllers, AD Connector uses the Active Directory site whose subnet IP address ranges are close to those in the VPC that contain the AD Connector. If you have a site whose subnets have the same IP address ranges as those in your VPC, AD Connector will discover the domain controllers in that site, which may not be physically close to your region.

Be Aware of Your Limits

By default, you are limited to 10 directories and 5 snapshots per each directory. You can increase those limits following the steps listed in AWS Directory Service Limits (p. 194).

Another limit you should pay attention to is number of users in a directory. Generally, you should not add more than 5,000 users to a Simple AD directory. If you have more than 5,000 users, consider AWS Directory Service for Microsoft Active Directory instead.

Understand Your Directory's AWS Security Group Configuration and Use

AWS creates a security group and attaches it to your directory's domain controller elastic network interfaces. This security group blocks unnecessary traffic to the domain controller and allows traffic that is necessary for Active Directory communications. AWS configures the security group to open only the ports that are required for Active Directory communications. In the default configuration, the security group accepts traffic to these ports from any IP address. AWS attaches the security group to your domain controllers' interfaces that are accessible from within your, peered or resized VPCs. These interfaces are inaccessible from the internet even if you modify routing tables, change the network connections to your VPC, and configure the NAT Gateway service. As such, only instances and computers that have a network path into the VPC can access the directory. This simplifies setup by eliminating the requirement for you to configure specific address ranges. Instead, you configure routes and security groups into the VPC that permit traffic only from trusted instances and computers.

Modifying the Directory Security Group

If you want to increase the security of your directories' security groups, you can modify them to accept traffic from a more restrictive list of IP addresses. For example, you could change the accepted addresses from 0.0.0.0/0 to a CIDR range that is specific to a single subnet or computer. Similarly, you might choose to restrict the destination addresses to which your domain controllers can communicate. Make such changes only if you fully understand how security group filtering works. For more information, see Amazon EC2 Security Groups for Linux Instances in the *Amazon EC2 User Guide*. Improper changes can result in loss of communications to intended computers and instances. AWS recommends that you do not attempt to open additional ports to the domain controller as this decreases the security of your directory. Please carefully review the AWS Shared Responsibility Model.

Warning
It is technically possible for you to associate the security groups, which your directory uses, with other EC2 instances that you create. However, AWS recommends against this practice. AWS may have reasons to modify the security group without notice to address functional or security needs of the managed directory. Such changes affect any instances with which you associate the directory security group. Furthermore, associating the directory security group with your EC2 instances creates a potential security risk for your EC2 instances. The directory security group accepts traffic on required Active Directory ports from any IP address. If you associate this Security Group with an EC2 instance that has a public IP address attached to the internet, then any computer on the internet can communicate with your EC2 instance on the opened ports.

Use Microsoft AD If Trusts Are Required

Simple AD does not support trust relationships. If you need to establish a trust between your AWS Directory Service directory and another directory, you should use AWS Directory Service for Microsoft Active Directory.

Setting Up: Creating Your Directory

Here are some suggestions to consider as you create your directory.

Remember Your Administrator ID and Password

When you set up your directory, you provide a password for the administrator account. That account ID is *Administrator* for Simple AD and *Admin* for Microsoft AD. Remember the password that you create for this account; otherwise you will not be able to add objects to your directory.

Create a DHCP Options Set

We recommend that you create a DHCP options set for your AWS Directory Service directory and assign the DHCP options set to the VPC that your directory is in. That way any instances in that VPC can point to the specified domain, and DNS servers can resolve their domain names.

For more information about DHCP options sets, see DHCP Options Set (p. 167).

Understand Username Restrictions for AWS Applications

AWS Directory Service provides support for most character formats that can be used in the construction of usernames. However, there are character restrictions that are enforced on usernames that will be used for signing in to AWS applications, such as Amazon WorkSpaces, Amazon WorkDocs, Amazon WorkMail, or Amazon QuickSight. These restrictions require that the following characters not be used:

- Spaces
- !"#$%&'()*+,/:;<=>?@[\]^`{|}~

Note
The @ symbol is allowed as long as it precedes a UPN suffix.

Using Your Directory

Here are some suggestions to keep in mind when using your directory.

Do Not Alter Predefined Users, Groups and Organization Units

When you use AWS Directory Service to launch a directory, AWS creates an organizational unit (OU) that contains all your directory's objects. This OU, which has the NetBIOS name that you typed when you created your directory, is located in the domain root. The domain root is owned and managed by AWS. Several groups and an administrative user are also created.

Do not move, delete or in any other way alter these predefined objects. Doing so can make your directory inaccessible by both yourself and AWS.

Automatically Join Domains

When launching a Windows instance that is to be part of an AWS Directory Service domain, it is often easiest to join the domain as part of the instance creation process rather than manually adding the instance later. To automatically join a domain, simply select the correct directory for **Domain join directory** when launching a new instance. You can find details in Launching an Instance (Simple AD and Microsoft AD) (p. 156).

Set Up Trusts Correctly

When setting up trust relationship between your Microsoft AD directory and another directory, keep in mind these guidelines:

- Both trusts must be forest trusts.
- Both fully qualified domain names (FQDNs) must be unique.
- If adding a NetBIOS name, that should also be unique.

For more details and specific instructions on setting up a trust relationship, see When to Create a Trust Relationship (p. 66).

Use Unique AD Connectors for Each Domain

AD Connectors and your on-premises domains have a 1-to-1 relationship. That is, for each on-premises domain you want to authenticate against, you must create a unique AD Connector.

Managing Your Directory

Consider these suggestions for managing your directory.

Make a Backup of Your Instance

If you decide to manually add an instance to an existing AWS Directory Service domain, make a backup or take a snapshot of that instance first. This is particularly important when joining a Linux instance. Some of the procedures used to add an instance, if not performed correctly, can render your instance unreachable or unusable.

Set Up SNS Messaging

With Amazon Simple Notification Service (Amazon SNS), you can receive email or text (SMS) messages when the status of your directory changes. You will be notified if your directory goes from an **Active** status to an **Impaired** or **Inoperable** status. You also receive a notification when the directory returns to an Active status.

Also remember that if you have an SNS topic that receives messages from AWS Directory Service, before deleting that topic from the Amazon SNS console, you should associate your directory with a different SNS topic. Otherwise you risk missing important directory status messages.

Remove Amazon RDS Databases before Deleting a Directory

Before deleting a directory that is associated with an Amazon Relational Database Service (Amazon RDS), you must first remove that database from the directory.

Amazon Cloud Directory

Amazon Cloud Directory is a highly available multi-tenant directory-based store in AWS. These directories scale automatically to hundreds of millions of objects as needed for applications. This lets operation's staff focus on developing and deploying applications that drive the business, not managing directory infrastructure. Unlike traditional directory systems, Cloud Directory does not limit organizing directory objects in a single fixed hierarchy.

With Cloud Directory, you can organize directory objects into multiple hierarchies to support many organizational pivots and relationships across directory information. For example, a directory of users may provide a hierarchical view based on reporting structure, location, and project affiliation. Similarly, a directory of devices may have multiple hierarchical views based on its manufacturer, current owner, and physical location.

At its core, Cloud Directory is a specialized graph-based directory store that provides a foundational building block for developers. With Cloud Directory, developers can do the following:

- Create directory-based applications easily and without having to worry about deployment, global scale, availability, and performance
- Build applications that provide user and group management, permissions or policy management, device registry, customer management, address books, and application or product catalogs
- Define new directory objects or extend existing types to meet their application needs, reducing the code they need to write
- Reduce the complexity of layering applications on top of Cloud Directory
- Manage the evolution of schema information over time, ensuring future compatibility for consumers

Cloud Directory includes a set of API operations to access various objects and policies stored in your Cloud Directory-based directories. For a list of available operations, see Amazon Cloud Directory API Actions. For a list of operations and the permissions required to perform each API action, see Amazon Cloud Directory API Permissions: Actions, Resources, and Conditions Reference (p. 183).

What Cloud Directory Is Not

Cloud Directory is not a directory service for IT Administrators who want to manage or migrate their directory infrastructure.

Read the topics in this section to get started creating directories, managing schemas and directory objects.

For additional resources, see Cloud Directory Resources (p. 53).

Understanding Key Cloud Directory Concepts

Amazon Cloud Directory is a directory-based data store that can create various types of objects in a schema-oriented fashion.

Directory

A directory is a schema-based data store that contains specific types of objects organized in a multi-hierarchical structure (see Directory Structure (p. 13) for more details). For example, a directory of users may provide a hierarchical view based on reporting structure, location, and project affiliation. Similarly, a directory of devices may have multiple hierarchical views based on its manufacturer, current owner, and physical location.

A directory defines the logical boundary for the data store, completely isolating it from all other directories in the service. It also defines the boundaries for an individual request. A single transaction or query executes within the context of a single directory. A directory cannot be created without a schema and typically has one schema applied to it. However, you can use the Cloud Directory API operations to apply additional schemas to a directory. For more information, see `ApplySchema` in the *Amazon Cloud Directory API Reference Guide*.

Objects

Objects are a structured data entity in a directory. An object in a directory is intended to capture metadata (or attributes) about a physical or logical entity usually for the purpose of information discovery and enforcing policies. For example users, devices, applications, AWS accounts, EC2 instances and Amazon S3 buckets can all be represented as different types of objects in a directory.

An object's structure and type information is expressed as a collection of facets. You can use `Path` or `ObjectIdentifier` to access objects. Objects can also have attributes, which are a user-defined unit of metadata. For example, the user object can have an attribute called *email-address*. Attributes are always associated with an object.

Policies

Policies are a specialized type of object that are useful for storing permissions or capabilities. Policies offer the `LookupPolicy` API action. The lookup policy action takes a reference to any object as its starting input. It then walks up the directory all the way to the root. The action collects any policy objects that it encounters on each path to the root. Cloud Directory does not interpret any of these policies in any way. Instead, Cloud Directory users interpret policies using their own specialized business logic.

For example, imagine a system that stores employee information. Employees are grouped together by job function. We want to establish different permissions for members of the Human Resources Group and the Accounting group. Members of the Human Resources group will have access to payroll information and the Accounting group will have access to ledger information. To establish these permissions, we attach policy objects to each of these groups. When it is time to evaluate a user's permissions, we can use the `LookupPolicy` API action on that user's object. The `LookupPolicy` API action walks the tree from the specified policy object up to the root. It stops at each node and checks for any attached policies and returns those.

Policy Attachments

Policies can be attached to other objects in two ways: normal parent-child attachments and special policy attachments. Using normal parent-child attachments, a policy can be attached to a parent node. This is often useful to provide an easy mechanism to locate policies within your data directory. Policies cannot have children. Policies attached via parent-child attachments will not be returned during `LookupPolicy` API calls.

Policy objects can also be attached to other objects via policy attachments. You can manage these policy attachments using the `AttachPolicy` and `DetachPolicy` API actions. Policy attachments allow policy nodes to be located when you use the LookupPolicy API.

Policy Schema Specification

In order to start using policies, you must first add a facet to your schema that support creating policies. To accomplish this, create a facet setting the `objectType` of the facet to POLICY. Creating objects using a facet with the type POLICY ensures that the object has policy capabilities.

Policy facets inherit two attributes in addition to any attributes you add to the definition:

- **policy_type** (String, Required) – This is an identifier you can provide to distinguish between different policy uses. If your policies logically fall into clear categories, we encourage setting the policy type attribute appropriately. The `LookupPolicy` API returns the policy type of attached policies (see `PolicyAttachment`). This allows easy filtering of the specific policy type that you are looking for. It also allows you to use policy_type to decide how the document should be processed or interpreted.
- **policy_document** (String, Required) – You can store application specific data in this attribute, such as permission grants associated with the policy. You can also store application-related data in normal attributes on your facet, if you prefer.

Policy API Overview

A variety of specialized API actions are available for working with policies. For a list of available operations, see Amazon Cloud Directory Actions.

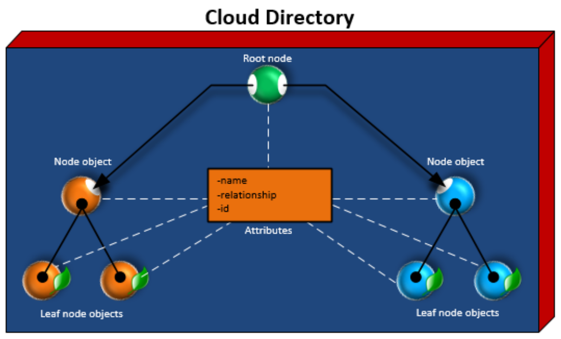
To create a policy object, use the `CreateObject` API action with an appropriate facet:

- To attach or detach a policy from an object, use the actions `AttachPolicy` and `DetachPolicy` respectively.
- To find policies that are attached to objects up the tree, use the `LookupPolicy` API action.
- To list the policies that are attached to a particular object, use the `ListObjectPolicies` API action.

For a list of operations and the permissions required to perform each API action, see Amazon Cloud Directory API Permissions: Actions, Resources, and Conditions Reference (p. 183).

Directory Structure

Data in a directory is structured hierarchically in a tree pattern consisting of nodes, leaf nodes, and links between the nodes, as shown in the following illustration. This is useful in application development to model, store, and quickly traverse hierarchical data.



Root Node

The root is the top node in a directory that is used to organize the parent and child nodes in the hierarchy. This is similar to how folders in a file system can contain subfolders and files.

Node

A node represents an object that can have child objects. For example, a node can logically represent a group of managers whereby various user objects are the children, or leaf nodes. A node object can only have one parent.

Leaf Node

A leaf node represents an object with no children that may or may not be directly connected to a parent node. For example, a user or device object. A leaf node object can have multiple parents. While leaf node objects are not required to be connected to a parent node, it is strongly recommended that you do so, since without a path from the root, the object can only be accessed by it's `NodeId`. If you misplace the id of such an Object, you will have no way to locate it again.

Node Link

The connection between one node and another. Cloud Directory supports a variety of link types between nodes, including parent-child links, policy links, and index attribute links.

Schema

A schema is a collection of facets that define what objects can be created in a directory and how they are organized. A schema also enforces data integrity and interoperability. A single schema can be applied to more than one directory at a time. For more information, see Schemas (p. 27).

Facet

A facet is a collection of attributes, constraints, and links defined within a schema. Combined together, facets define the objects in a directory. For example, Person and Device can be facets to define corporate employees with association of multiple devices. For more information, see Facets (p. 30).

Sample Schemas

The set of sample schemas provided by default in the AWS Directory Service console. For example, Person, Organization, and Device are all sample schemas. For more information, see Sample Schemas (p. 31).

Custom Schemas

One or more schemas defined by a user that can be uploaded from the Schemas section or during the Cloud Directory creation process of the AWS Directory Service console, or created by API calls.

Using the Console

With Amazon Cloud Directory, the AWS Directory Service console lets you do the following:

- Create, and delete directories
- View directory details
- Create, upload, and delete schemas
- View schema details

Topics

Create an Amazon Cloud Directory

Before you can create a directory in Amazon Cloud Directory, AWS Directory Service requires that you first apply a schema to it. A directory cannot be created without a schema and typically has one schema applied to it. However, you use Cloud Directory API operations to apply additional schemas to a directory. For more information, see ApplySchema in the *Amazon Cloud Directory API Reference Guide*.

To create a Cloud Directory

1. In the AWS Directory Service console navigation pane, select **Directories** and choose **Set up directory**.
2. Choose **Amazon Cloud Directory**.
3. For **Name**, type the friendly name of your directory, such as `User Repository`.
4. Under **Choose or add a new schema**, choose a sample schema from the list or choose **Upload new schema (JSON)** to upload a custom schema, and then choose **Next**. Currently, Cloud Directory provides the following Sample Schemas (p. 31):

   - Organization
   - Person (User)
   - Device

   Sample schemas and newly uploaded schemas are placed in the **In Development** state, by default. For more information about schema states, see Schema Lifecycle (p. 28). Before a schema can be applied to a directory, it must be converted into the **Published** state.

   To successfully publish an AWS sample schema using the console, you must have permissions to the following actions:

   - `clouddirectory:Get*`
   - `clouddirectory:List*`
   - `clouddirectory:CreateSchema`
   - `clouddirectory:CreateDirectory`
   - `clouddirectory:PutSchemaFromJson`
   - `clouddirectory:PublishSchema`
   - `clouddirectory:DeleteSchema`

   Since sample schemas are read-only templates provided by AWS, they cannot be published directly. Instead, when you choose to create a directory based on a sample schema, the console creates a temporary copy of the sample schema you selected and places it in the **In Development** state. It then creates a copy of that development schema and places it in the **Published** state. Once published, the development schema is deleted, which is why the `DeleteSchema` action is necessary when publishing a sample schema.
5. Review the directory information and make any necessary changes. When the information is correct, choose **Launch**.

Create a Schema

Amazon Cloud Directory supports uploading of a compliant JSON file for schema creation. You can also create, read, update, and delete schemas using the Cloud Directory APIs. For more information about schema API operations, see the Amazon Cloud Directory API Reference Guide.

To create a new schema, you can either create your own JSON file from scratch or download one of the existing schemas listed in the console. Then upload it as a custom schema. For more information, see Custom Schemas (p. 35). Choose either of the procedures below, depending on your preferred method.

To create a custom schema

1. In the AWS Directory Service console navigation pane, choose **Schemas**.
2. Create a JSON file with all of your new schema definitions. For more information about how to format a JSON file, see JSON Schema Format (p. 36).
3. In the console, choose **Upload new schema (JSON)**, select the new JSON file that you just created, and then choose **OK**.

   This adds a new schema to your schema library and place it in the **In Development** state. For more information about schema states, see Schema Lifecycle (p. 28).

To create a custom schema based on an existing one in the console

1. In the AWS Directory Service console navigation pane, select **Schemas**.
2. In the table listing the schemas, select the schema you want to edit, and then choose **Download schema**.
3. Rename the JSON file, edit it as needed, and then save the file. For more information about how to format a JSON file, see JSON Schema Format (p. 36).
4. In the console, choose **Upload new schema (JSON)**, select the JSON file that you just edited, and then choose **OK**.

   This adds a new schema to your schema library and place it in the **In Development** state. For more information about schema states, see Schema Lifecycle (p. 28).

Directory Objects

Developers model directory objects using extensible schemas to enforce data correctness constraints automatically, making it easier to program for. Amazon Cloud Directory offers rich information lookup based on your defined indexed attributes, thus enabling fast tree traversals and searches within the directory trees. Cloud Directory data is encrypted at rest and in transit.

Topics

Objects and Links

This section describes objects and links and how the directory identifies them.

Topics

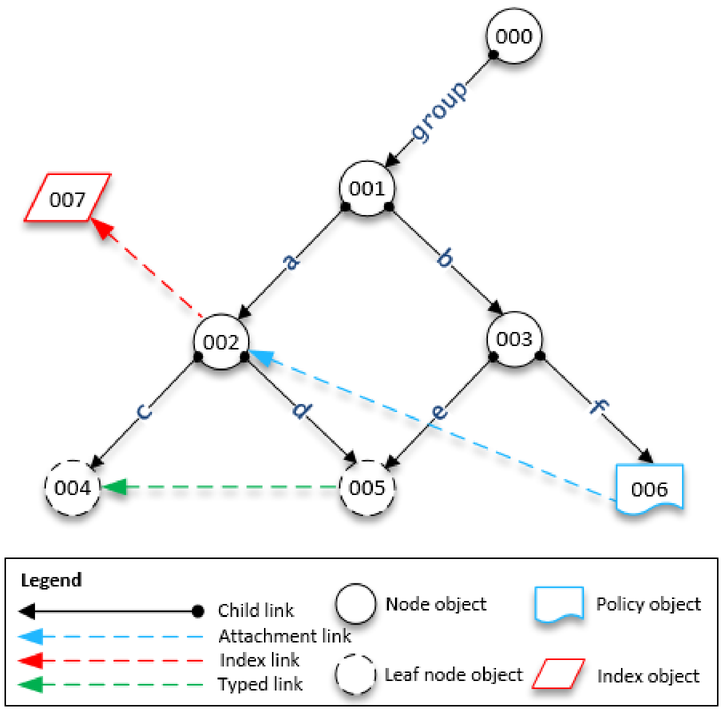
Objects

An object is a basic element of Cloud Directory. Each object has a globally unique identifier, which is specified by the object Identifier. An object is a collection of zero or more facets with their attribute keys and values. An object can be created from one or more facets within a single applied schema or from facets of multiple applied schemas. During object creation, you must specify all required attribute values. Objects can have a limited number of facets. For more information, see AWS Directory Service Limits (p. 194).

An object can be a regular object, a policy object, or an index object. An object can also be a node object or a leaf node object. The type of the object is inferred from the object type of the facets attached to it.

Links

A link is a directed edge between two objects that define a relationship. Cloud Directory currently supports the following link types.



Child Links

A child link creates a parent–child relationship between the objects it connects. For example, in the above illustration child link **b** connects objects `001` and `003`. Child links define the hierarchy in Cloud Directory. Child links have names when they participate in defining the path of the object that the link points to.

Attachment Links

An attachment link applies a leaf node policy object to another leaf node or a node object. Attachment links do not define the hierarchical structure of Cloud Directory. For example, in the above illustration, attachment link applies the policy stored in policy leaf node object `006` on node object `002` . Each object can have multiple policies attached but not more than one policy of any given policy type can be attached.

Index Links

Index links provide rich information lookup based on an index object and your defined indexed attributes, thus enabling fast tree traversals and searches within the directory trees. Conceptually, indexes are similar to nodes with children: The links to the indexed nodes are labeled according to the indexed attributes, rather than being given a label when the child is attached. However, index links are not parent-child edges, and have their own set of enumeration API operations. For more information, see Indexing and Search (p. 42).

Typed Links

Typed links are a special type of link. With a typed link, you can attach any two objects and add attributes to that attachment. You can use typed links to model relationships between different objects in your directory. For example, in the illustration above, consider the relationship between object `004`, which represents a user, and object `005`, which represents a device. We might use a typed link to model an ownership relationship between the two objects. We could add attributes to the typed link to represent the date of purchase or whether the device is rented or purchased.

As with objects, you must create a typed link facet using the `CreateTypedLinkFacet` API to define the typed link structure and its attributes. Typed link facets require a unique facet name and set of attributes that are associated with the link. With Cloud Directory, the schema designer can define an ordered set of attributes on the typed link facet. To view a typed links sample schema, see Schema Document with Typed Links (p. 39).

Typed link attributes can be used when you need to do any of the following:

- Represent the relationship between two objects.
- Allow for quick filtering of incoming or outgoing typed links. For more information, see Typed Link Listing (p. 19).

Consider the following when deciding if typed links are right for your use case:

- Typed links cannot be used in path-based object specification. Instead, you must select typed links using the `ListOutgoingTypedLinks` or `ListIncomingTypedLinks` API operations.
- Typed links do not participate in `LookupPolicy` or `ListObjectParentPaths` API operations.
- Typed links between the same two objects and in the same direction may not have the same attribute values. This can help avoid duplicated typed links between the same objects.
- The combined size of all identity attribute values is limited to 64 bytes. For more information, see AWS Directory Service Limits (p. 194).

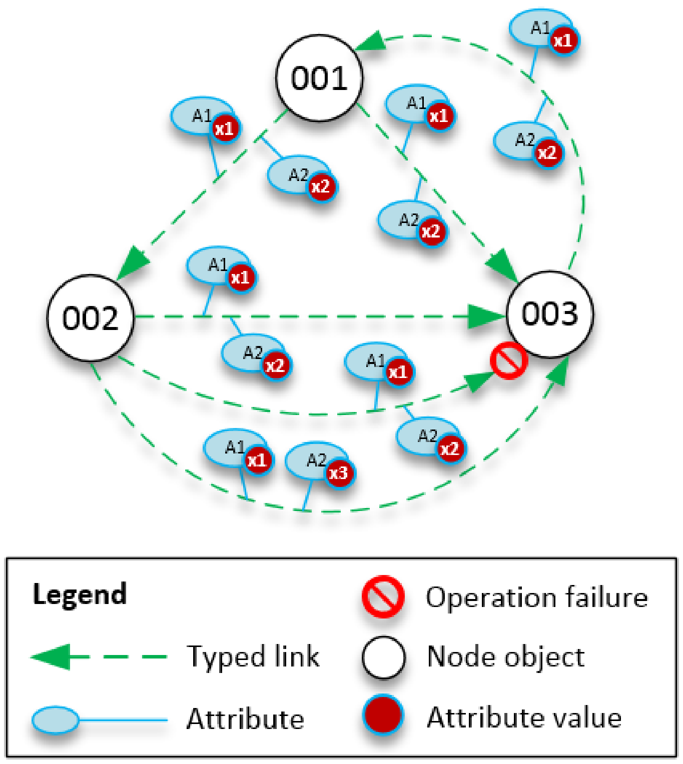
Related Cloud Directory Blog Article

- Use Amazon Cloud Directory Typed Links to Create and Search Relationships Across Hierarchies

Typed Link Identity

Identity is what uniquely defines whether a typed link can exist between two objects. The exception is when you connect two objects in one direction with the exact same attribute values. Attributes must be configured as `REQUIRED_ALWAYS`, since all attributes contribute to identity.

Typed links that are created from different typed link facets never conflict with each other. For example, consider the following diagram:



- Object `001` has typed links and attributes (A1 and A2) with the same attribute values (x1 and x2) going to different objects (`002` and `003`). This operation would succeed.
- Objects `002` and `003` have a typed link between them. This operation would fail because two typed links in the same direction with the same attributes cannot exist between objects.
- Objects `001` and `003` have two typed links between them with the same attributes. However, since the links go in different directions, this operation would succeed.
- Objects `002` and `003` have typed links between them with the same value for A1 but different values for A2. Typed link identity considers all attributes so this operation would succeed.

Typed Link Rules

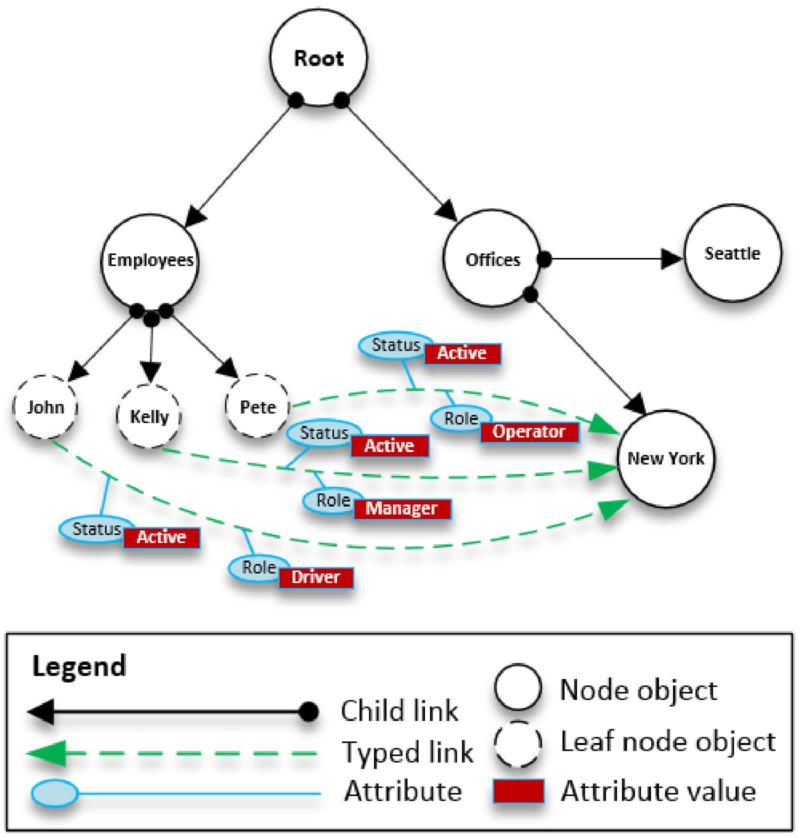
You can add rules to typed link attributes when you want to add restrictions to link attributes. These rules are equivalent to rules on object attributes. For more information, see Attribute Rules (p. 35).

Typed Link Listing

Cloud Directory provides API operations that you can use to select incoming or outgoing typed links from an object. You can select a specific subset of typed links rather than iterating over every typed link. You can also specify a particular typed link facet to filter only typed links of that type.

You can filter typed links based on the order that the attributes are defined on the typed link facet. You can provide range filters for multiple attributes. When providing ranges to a typed link selection, any inexact ranges much be specified at the end. Any attributes with no range specified are presumed to match the entire range. Filters are interpreted in the order of the attributes that are defined on the typed link facet, not the order they are supplied to any API calls.

For example, in the following diagram, consider a Cloud Directory that is used to store information about Employees and their Abilities.



Let's say we model our employee's capabilities with a typed link named `EmployeeCapability`, which is configured with two string attributes: `Status` and `Role`. The following filters are supported on `ListIncomingTypedLinks` and `ListOutgoingTypedLinks` API operations.

- Facet = `EmployeeCapability`, Status = `Active`, Role = `Driver`
  - Selects active employees who are drivers. This filter includes two exact matches.
- Facet = `EmployeeCapability`, Status = `Active`
  - Selects all active employees.
- Facet = `EmployeeCapability`, Status = `Active`, Role = `A` to `M`
  - Selects active employees with roles starting with `A` through `M`.
- Facet = `EmployeeCapability`
  - This selects all typed links of the `EmployeeCapability` type.

The following filters would **NOT** be supported:

- Facet = `EmployeeCapability`, Status between `A` to `C`, Role = `Driver`
  - This filter is not allowed because any ranges must appear at the end of the filter.
- Facet = `EmployeeCapability`, Role = `Driver`
  - This filter is not allowed because the implicit status range is not an exact match and does not appear at the end of the list of ranges.
- Status = `Active`
  - This filter is not allowed because the typed link facet is not specified.

Typed Link Schema

You can create typed link facets in two ways. You can manage your typed link facets from individual API calls, including `CreateTypedLinkFacet`, `DeleteTypedLinkFacet`, and `UpdateTypedLinkFacet`. You can also upload a JSON document that represents your schema in a single `PutSchemaFromJson` API call. For more information, see JSON Schema Format (p. 36). To view a typed links sample schema, see Schema Document with Typed Links (p. 39).

The types of changes allowed at different phases of the schema development lifecycle are similar to changes that are allowed for object facet manipulation. Schemas in the development state support any changes. Schemas in the published state are immutable and no changes are supported. Only certain changes are allowed to schemas that are applied to a data directory. Once you set the order and attributes on an applied typed link facet, that order cannot be changed.

Two other API operations list facets and their attributes:

- `ListTypedLinkFacetAttributes`
- `ListTypedLinkFacetNames`

Typed Link Interaction

Once a typed link facet has been created, you are ready to start creating and interacting with typed links. To attach and detach typed links, use the `AttachTypedLink` and `DetachTypedLink` API operations.

The `TypedLinkSpecifier` is a structure that contains all the information to uniquely identify a typed link. Within that structure you can find `TypedLinkFacet`, `SourceObjectID`, `DestinationObjectID`, and `IdentityAttributeValues`. These are used to uniquely specify the typed link being operated on. The `AttachTypedLink` API operation returns a typed link specifier while the `DetachTypedLink` API operation accepts one as input. Similarly, the `ListIncomingTypedLinks` and `ListOutgoingTypedLinks` API operations provide typed link specifiers as output. You can construct a typed link specifier from scratch as well. The full list of typed link-related API operations, include the following:

- `AttachTypedLink`
- `CreateTypedLinkFacet`
- `DeleteTypedLinkFacet`
- `DetachTypedLink`
- `GetTypedLinkFacetInformation`
- `ListIncomingTypedLinks`
- `ListOutgoingTypedLinks`
- `ListTypedLinkFacetNames`
- `ListTypedLinkFacetAttributes`
- `UpdateTypedLinkFacet`

Note

- Attribute references and updating typed links are not supported. To update a typed link, you must remove it and add the updated version.
- Typed links cannot contain attributes that do not contribute to identity.
- Batch support for typed links is not yet supported but is planned for release in the near future.

Range Filters

Several Cloud Directory list APIs allow specifying a filter in the form of a range. These filters allow you to efficiently select subsets of the links attached to the specified node.

Ranges are generally supplied as a map (array of key-value pairs) whose keys are attribute identifiers and whose values are the corresponding ranges. This allows filtering links whose identities consist of one or more attributes. For example, a TypedLink set up to model a Role relationship for determining permissions might have both RoleType and Authorizer attributes. A `ListOutgoingTypedLinks` call could then specify ranges to filter the result to RoleType:"Admin" and Authorizer:"Julia". The map of ranges used to filter a single list request must contain only attributes that define the link's identity (an index's OrderedIndexedAttributeList or a TypedLink's IdentityAttributeOrder), but it need not contain ranges for all of them. Missing ranges will automatically be filled in with ranges that span all possible values (from FIRST to LAST).

If you think of each attribute as defining an independent flat domain of values, the range structures define two logical points in that domain — the start and end points — and the range matches all of the possible points in between those points. The StartValue and EndValue of the range structure define the basis for these two points with the "modes" further refining them to indicating whether each point itself is to be included or excluded from the range. In the RoleType:"Admin" example above, the values for the RoleType attribute would both be "Admin", and the modes are both "INCLUSIVE" (written as ["Admin" to "Admin"]). A filter for a ListIndex call where the index is defined on the LastName of a User facet might use StartValue="D", StartMode=INCLUSIVE, EndValue:"G", EndMode:EXCLUSIVE to narrow the listing to names starting with D, E, or F.

The start point of a range must always precede or be equal to the end point. Cloud Directory will return an error if EndValue precedes the StartValue. The values must also be of the same primitive type as the attribute they are filtering, String values for a String attribute, Integer for an Integer attribute, and so on. StartValue="D", StartMode=EXCLUSIVE, EndValue="D", EndMode=INCLUSIVE is invalid, for instance, because the end point includes the value while the start point follows the value.

There are three special modes that can be used by either start or end points. The following modes do not require the corresponding value field to be specified, as they imply a position on their own.

- **FIRST -** precedes all possible values in the domain. When used for the start point, this matches all possible values from the beginning of the domain up to the end point. When used for the end point, no values in the domain will match the range.
- **LAST -** follows all possible values in the domain. When used for the end point, this matches all possible values that follow the start point, including missing values. When used for the start point, no values in the domain will match the range.
- **LAST_BEFORE_MISSING_VALUES -** This mode is only useful for optional attributes where the value may be omitted (see Missing values (p. 23)). It corresponds to the point between the missing values and the actual domain values. When used for the end point, this matches all non-missing domain values that follow the start point. When used for the start point, it excludes all non-missing domain values. If the attribute is a required one, this mode is equivalent to LAST, as there can be no missing values.

Multiple range limitations

Cloud Directory limits patterns where there are multiple attributes in order to guarantee efficient, low-latency request processing. Each link with multiple identifying attributes specifies them in a well-defined order. For instance, the Role example above defines the RoleType attribute as most significant, and the Authorizer attribute as least significant. A List request can specify only a single "qualifying" range that is not either 1) a single value or 2) spans all possible values (there can be multiple ranges that match these two requirements). Any ranges for more significant attributes than the qualifying range attribute must specify a single value, and any ranges for less significant ranges must span all possible values. In the Role example, the filter sets (RoleType:"Admin", Authorizer:["J" to "L"]) (single value + qualifying range), (RoleType:["Admin" to "User"]) (qualifying range + implicit spanning range), and (RoleType:[FIRST to LAST]) (two spanning ranges, one implicit) are all examples of valid filter sets. (RoleType:[FIRST to LAST], Authorizer:"Julia") is not a valid set, since the spanning range is more significant than the single value range.

Some useful patterns when filling in the range structures, include:

Matching a single value

Specify the value for both StartValue and EndValue, and set both modes to "INCLUSIVE".

Example: `StartValue="Admin", StartMode=INCLUSIVE, EndValue="Admin", EndMode=INCLUSIVE`

Matching a prefix

Specify the prefix as the StartValue with INCLUSIVE mode, and the first value after the prefix as EndValue with an EXCLUSIVE mode.

Example: `StartValue="Jo", StartMode=INCLUSIVE, EndValue="Jp", EndMode=EXCLUSIVE ("p" is the next character value after "o")`

Filtering for Greater Than a value

Specify the value for StartValue with EXCLUSIVE mode, and LAST as the EndMode (or LAST_BEFORE_MISSING_VALUES to exclude missing values, if applicable).

Example: `StartValue=127, StartMode=EXCLUSIVE, EndValue=null, EndMode=LAST`

Filtering for Less Than or equal to a value

Specify the value for the EndValue with INCLUSIVE mode, and FIRST as the StartMode.

Missing values

When an attribute is marked as Optional in the schema, it's value may be "missing" since it need not have been supplied when the facet was attached or the attribute could have been subsequently deleted. If the object with such a missing value is attached to an index, the index link is still present, but moved to the end of the set of links. A `ListIndex` call will first return any links where the indexed attributes are all present before returning links where one or more are missing. This is roughly similar to a relational database NULL value, with these values ordered after non-NULL values. You can specify whether a range includes these missing values or not by choosing the LAST or LAST_BEFORE_MISSING_VALUES modes. For example, you provide a filter to a ListIndex call to return just the missing values in an index by filtering with the range [LAST_BEFORE_MISSING_VALUES to LAST].

Accessing Objects

Objects in a directory can be accessed either by path or by `objectIdentifier`.

Path: Every object in a Cloud Directory tree can be identified and found by the pathname that describes how to reach it. The path starts from the root of the directory (Node `000` in the previous figure). The path notation begins with the link labeled with a slash (/) and follows the child links separated by path separator (also a slash) until reaching the last part of the path. For example, object `005` in the previous figure can be identified using the path `/group/a/d`. Multiple paths may identify an object, since objects that are leaf nodes can have multiple parents. The following path can also be used to identify object `005` : /group/b/e

ObjectIdentifier: Every object in the directory has a unique global identifier, which is the `ObjectIdentifier`. `ObjectIdentifier` is returned as part of the `CreateObject` API call. You can also fetch the `ObjectIdentifier` by using `GetObjectInformation` API call. For example, to fetch the object identifier of object `005`, you can call `GetObjectInformation` with object reference as the path that leads to the object, which is group/b/e or group/a/d.

```
GetObjectInformationRequest request = new GetObjectInformationRequest()
    .withDirectoryArn(directoryArn)
    .withObjectReference("/group/b/e")
    .withConsistencyLevel(level)
GetObjectInformationResult result = cdClient.getObjectInformation(request)
String objectIdentifier = result.getObjectIdentifier()
```

Populating Objects

New facets can be added to an object using `AddFacetToObject` API call. The type of the object is determined based on the facets attached to the object. Object attachment in a directory works based on the type of the object. When attaching an object, remember these rules:

- A leaf node object cannot have children.
- A node object can have multiple children.
- An object of the policy type cannot have children and can have zero or one parent.

Updating Objects

You can update an object in multiple ways:

1. Use the `UpdateObjectAttributes` operation to update individual facet attributes on an object.
2. Use the `AddFacetToObject` operation to add new facets to an object.
3. Use the `RemoveFacetFromObject` operation to delete existing facets from an object.

Deleting Objects

An attached object must meet certain conditions before you can delete it from a directory:

1. You must detach the object from the tree. You can detach an object only when it doesn't have any children. If the object has children, you must detach all the children first.
2. You can delete a detached object only if all the attributes on that object are deleted. You can delete attributes on an object by deleting each facet attached to that object. You can fetch a list of facets attached to an object by calling `GetObjectInformation`.
3. An object must also have no parent, no policy attachments, and no index attachments.

Because an object must be fully detached from the tree to be deleted, you must use the object identifier to delete it.

Querying Objects

This section discusses various elements relevant for querying objects in a directory.

Directory Traversal

Because Cloud Directory is a tree, you can query objects from the top down using the `ListObjectChildren` API operation or from the bottom up using the `ListObjectParents` API operation.

Policy Lookup

Given an object reference, the `LookupPolicy` API operation returns all the policies that are attached along its path (or paths) to the root in a top-down fashion. Any of the paths that are not leading up to the root are ignored. All policy type objects are returned.

If the object is a leaf node, it can have multiple paths to the root. This call returns only one path for each call. To fetch additional paths, use the pagination token.

Index Querying

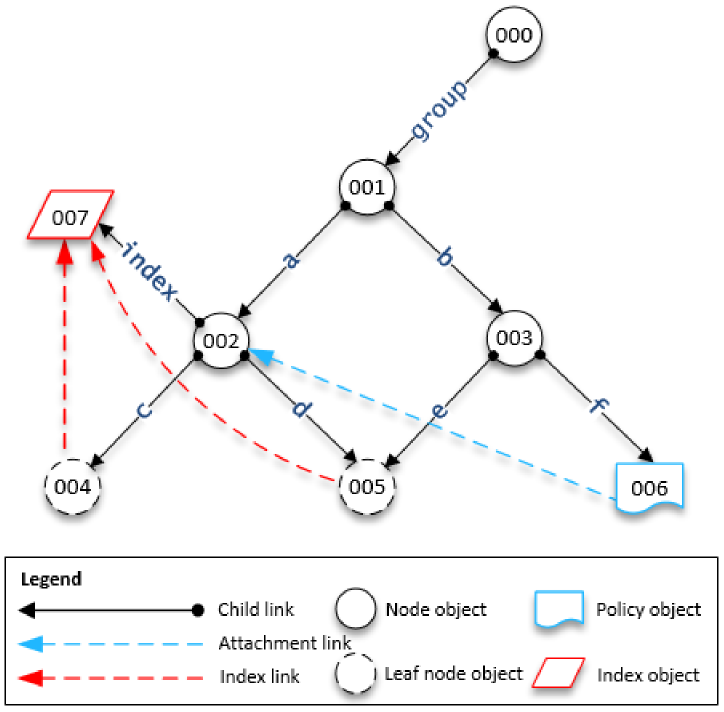
Cloud Directory supports rich index querying functionality with the use of the following ranges:

- **FIRST -** Starts from the first indexed attribute value. The start attribute value is optional.
- **LAST -** Returns attribute values up to the end of the index, including the missing values. The end attribute value is optional.
- **LAST_BEFORE_MISSING_VALUES -** Returns attribute values up to the end of index, excluding missing values.
- **INCLUSIVE -** Includes the attribute value being specified.
- **EXCLUSIVE -** Excludes the attribute value being specified.

Parent Path Listing

Using the `ListObjectParentPaths` API call, you can retrieve all available parent paths for any type of object (node, leaf node, policy node, index node). This API operation can be helpful when you need to evaluate all parents for an object. The call returns all the objects from the directory root until the requested object. It also returns the number of paths based on user-defined `MaxResults`, in case of multiple paths to the parent. The order of the paths and nodes returned is consistent among multiple API calls unless the objects are deleted or moved. Paths not leading to the directory root are ignored from the target object.

For an example on how this works, let's say a directory has an object hierarchy similar to the illustration shown below.



The numbered shapes represent the different objects. The number of arrows between that object and the directory root (`000`) represent the complete path and would be expressed in the output. The following table shows requests and responses from queries made to specific leaf node objects in the hierarchy.

Example queries on objects

Request	Response
`004, PageToken: null, MaxResults: 1`	`[{/group/a/c], [000, 001, 002, 004]}], PageToken: null`
`005, PageToken: null, MaxResults: 2`	`[{/group/a/d, [000, 001, 002, 005]}, { /group/b/e, [000, 001, 003, 005]}], PageToken: null` **Note** In this example, object `005` has both nodes `002` and `003` as parents. Also, since `MaxResults` is 2, both paths display objects in a list.
`005, PageToken: null, MaxResults: 1`	`[{/group/a/d, [000, 001, 002, 005]}], PageToken: <encrypted_next_token>`
`005, PageToken: <encrypted_next_token>, MaxResults: 1`	`[{/group/b/e, [000, 001, 003, 005]}], PageToken: null` **Note** In this example, object `005` has both nodes `002` and `003` as parents. Also, since `MaxResults` is 1, multiple paginated calls with page tokens will be made to get all paths with a list of objects.
`006, PageToken: null, MaxResults: 1`	`[{/group/b/f, [000, 001, 003, 006]}], PageToken: null`
`007, PageToken: null, MaxResults: 1`	`[{/group/a/index, [000, 001, 002, 007]}], PageToken: null`

Consistency Levels

Amazon Cloud Directory is a distributed directory store. Data is distributed to multiple servers in different Availability Zones. A successful write request updates the data on all servers. Data is eventually available on all servers, usually within a second. To aid users of the service, Cloud Directory offers two consistency levels for read operations. This section describes the different consistency levels and eventually consistent nature of Cloud Directory.

Read Isolation Levels

When reading data from Cloud Directory, you must specify the isolation level you want to read from. Different isolation levels have tradeoffs between latency and data freshness.

- **EVENTUAL -** The snapshot isolation level reads whatever data is immediately available**.** It provides the lowest latency of any isolation level. It also provides a potentially old view of the data in the directory. EVENTUAL isolation does not provide read-after-write consistency. This means it is not guaranteed that you will be able to read data immediately after writing it.
- **SERIALIZABLE -** The serializable isolation level provides the highest level of consistency offered by Cloud Directory. Reads done at the SERIALIZABLE isolation level ensure that you receive data from any successful writes. If a change has been made to the data that you requested and that change is not yet available, the system rejects your request with `RetryableConflictException`. We recommend that you retry these exceptions (see the following section). When successfully retried, SERIALIZABLE reads offer read-after-write consistency.

Write Requests

Cloud Directory ensures that multiple write requests are not concurrently updating the same object or objects. If two write requests are found to be operating on the same objects, one of the operations fails with a `RetryableConflictException`. We recommend that you retry these exceptions (see the section below).

> **Note**
> `RetryableConflictException` responses received during write operations cannot be used to detect race conditions. Given a use case that has been shown to precipitate this situation, there is no guarantee that an exception will always occur. Whether an exception occurs or not depends on the order of each request being processed internally.

RetryableConflictExceptions

When performing write operations or read operations with a SERIALIZABLE isolation level after a write on the same object, Cloud Directory may respond with a `RetryableConflictException`. This exception indicates that Cloud Directory servers have not yet processed the contents of the previous write. These situations are transient and remedy themselves quickly. It's important to note that the `RetryableConflictException` cannot be used to detect any type of read-after-write consistency. There is no guarantee a particular use case will cause this exception.

We recommend that you configure your Cloud Directory clients to retry the `RetryableConflictException`. This configuration provides error-free behavior during operation. The following sample code demonstrates how this configuration can be made in Java.

```
      RetryPolicy retryPolicy = new RetryPolicy(new CloudDirectoryRetryCondition(),
          PredefinedRetryPolicies.DEFAULT_BACKOFF_STRATEGY,
          PredefinedRetryPolicies.DEFAULT_MAX_ERROR_RETRY,
                true);

       ClientConfiguration clientConfiguration = new
ClientConfiguration().withRetryPolicy(retryPolicy);


 AmazonCloudDirectory client = new AmazonCloudDirectory (
   new BasicAWSCredentials(…), clientConfiguration);


public static class CloudDirectoryRetryCondition extends SDKDefaultRetryCondition {

 @Override
 public boolean shouldRetry(AmazonWebServiceRequest originalRequest, AmazonClientException
exception,
   int retriesAttempted) {

  if (exception.getCause() instanceof RetryableConflictException) {
   return true;
  }

  return super.shouldRetry(originalRequest, exception, retriesAttempted);
 }
}
```

Schemas

With Amazon Cloud Directory, schemas define what types of objects can be created within a directory (users, devices, and organizations), enforce validation of data for each object class, and handle changes to the schema over time. More specifically, a schema defines the following:

- One or more types of facets that may be mapped to objects within a directory (such as Person, Organization_Person)
- Attributes that may be mapped to objects within a directory (such as Name, Description). Attributes can be required or made optional on various types of facets, and are defined within the context of a facet.
- Constraints that may be enforced on object attributes (such as Required, Integer, String)

When a schema has been applied to a directory, all data within that directory must then conform to that applied schema. In this way, the schema definition is essentially a blueprint that can be used to construct multiple directories with applied schemas. Once built, those applied schemas may vary from the original blueprint, each in different ways.

Applied schemas can later be updated using versioning and then reapplied to all the directories that use it. For more information, see In-Place Schema Upgrade (p. 30).

Cloud Directory provides API operations to create, read, update, and delete schemas. This allows the contents of the schema to be easily consumed by programmatic agents. Such agents access the directory to discover the full set of facets, attributes, and constraints that apply to data within the directory. For more information about the schema APIs, see the Amazon Cloud Directory API Reference Guide.

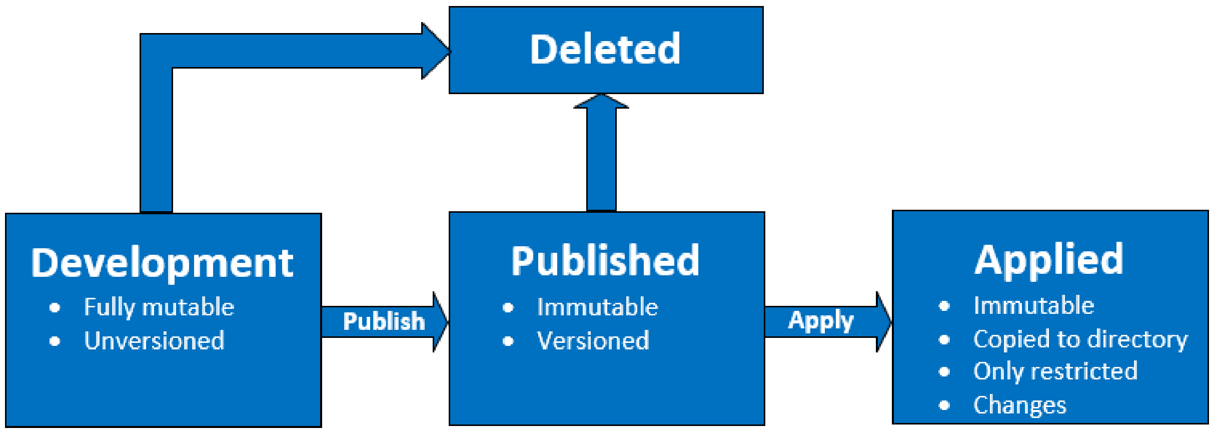
Cloud Directory supports uploading a compliant JSON file for schema creation. You can also create and manage schemas using the AWS Directory Services console. For more information, see Create an Amazon Cloud Directory (p. 15).

Topics

Schema Lifecycle

Cloud Directory offers a schema lifecycle to help with the development of schemas. This lifecycle consists of three states: Development, Published, and Applied. These states are designed to facilitate construction and distribution of schemas. Each of these states has different features aiding this effort.

The following diagram depicts possible transitions and verbiage. All schema transitions are copy-on-write. For example, publishing a development schema does not alter or remove the development schema.



You can delete a schema when it is in either the Development or Published state. Deleting a schema cannot be undone nor can it be restored once it has been deleted.

Schemas in Development, Published and Applied states have ARNs that represent them. These ARNs are used in API operations to describe the schema that the API operates on. It is easy to discern the state of a schema by looking at a schema ARN.

- Development: `arn:aws:clouddirectory:us-east-1:1234567890:schema/development/`*`SchemaName`*
- Published: `arn:aws:clouddirectory:us-east-1:1234567890:schema/published/`*`SchemaName`*`/`*`Version`*
- Applied: `arn:aws:clouddirectory:us-east-1:1234567890:directory/directoryid/schema/`*`SchemaName`*`/`*`Version`*

Development State

Schemas are initially created in the development state. Schemas in this state are fully mutable. You can freely add or remove facets and attributes. The majority of schema design occurs in this state. Schemas in this state have a name but no version.

Published State

The published schema state stores schemas that are ready to be applied to data directories. Schemas are published from the development state into the published state. You cannot change schemas in the published state. You can apply published schemas to any number of data directories.

Published and applied schemas must have a version associated with them. For more information about versions, see Schema Versioning (p. 30).

Applied State

A published schema can be applied to data directories. A schema that has been applied to a data directory is said to be applied. Once you apply a schema to a data directory, you can use the schema's facets when creating objects. You can apply multiple schemas to the same data directory. Only the following changes are permitted on an applied schema.

- Add a facet to an applied schema
- Add a non-required attribute to an applied schema

Facets

Facets are the most basic abstraction within a schema. They represent a set of attributes that can be associated with an object in the directory and are similar in concept to LDAP object classes. Each directory object may have up to a certain number of facets associated with it. For more information, see AWS Directory Service Limits (p. 194).

Each facet maintains its own independent set of attributes. Each facet consists of fundamental metadata, such as the facet name, version information, and behaviors. The combination of schema ARNs, facets, and attributes define uniqueness on the object.

The set of object facets, their constraints, and the relationships between them constitute an abstract schema definition. Schema facets are used to define constraints over the following things:

1. Attributes allowed in an object
2. Policy types allowed to apply to an object

Once you have added the necessary facets to your schema, you can apply the schema to your directory and create the applicable objects. For example, you can define a device schema by adding facets such as computers, phones, and tablets. You can then use these facets to create computer objects, phone objects, and tablet objects in the directory to which the schema applies.

Cloud Directory's schema support makes it easy to add or modify facets and attributes without worrying about breaking applications. For more information, see In-Place Schema Upgrade (p. 30).

In-Place Schema Upgrade

Cloud Directory offers the updating of existing schema attributes and facets to help integrate your applications with AWS provided services. Schemas that are in either the published or applied states have versions and cannot be changed. For more information, see Schema Lifecycle (p. 28).

Schema Versioning

A schema version indicates a unique identifier for a schema that developers can specify when programming their applications to conform to certain rules and formatting of data. Two key differentiators in the way versioning works with Cloud Directory are important for developers to understand. These differentiators—major version and minor version—can determine how future schema upgrades impact your application.

Major Version

Major version is the version identifier used for tracking major version changes for a schema. It can be up to 10 characters in length. Different versions of the same schema are completely independent. For example, two schemas with the same name and different versions are treated as completely different schemas, which have their own namespaces.

Backward incompatible changes

We recommend making changes to the major version only when schemas are incompatible. For example, when changing the data type of an existing attribute (such as changing from `string` to `integer`) or dropping a mandatory attribute from your schema. Backward-incompatible changes require directory data migration from a previous schema version to the new schema version.

Note
Until December 2017, Cloud Directory has referred to the major version identifier as *version*. Moving forward, we will refer to it as *major version*.

Minor Version

Minor version is the version identifier used for in-place upgrading of schemas or when you want to make backward-compatible upgrades such as adding additional attributes or adding facets. An upgraded schema using a minor version can be applied in place across all directories that use it without breaking any running applications. This includes directories that are used in production environments. For an example use case, see "How to Easily Apply Amazon Cloud Directory Schema Changes with In-Place Schema Upgrades" in the Cloud Directory Blog.

The minor version information and history is saved along with the other schema information in the schema metadata repository. Cloud Directory retains up to five of the most recent minor versions. No minor version information is retained in the objects. The advantage of introducing minor version is that client code works seamlessly as long as the major version is not changed.

Using the Schema Upgrade API Operations

You can use the `UpgradePublishedSchema` API call to upgrade published schemas. Schema upgrades are applied in place to the directories that rely on it using the `UpgradeAppliedSchema` API call. You can also fetch the major and minor version of an applied schema by calling `GetAppliedSchemaVersion`s. Or view the associated schema ARNs and schema revision history for a directory by calling `ListAppliedSchemaArns`. Cloud Directory maintains the five most recent versions of applied schema changes.

For an illustrative example, see "How to Easily Apply Amazon Cloud Directory Schema Changes with In-Place Schema Upgrades" in the Cloud Directory Blog. The blog post will demonstrate how you perform an in-place schema upgrade and use schema versions in Cloud Directory. It covers how to add additional attributes to an existing facet, add a new facet to a schema, publish the new schema, and apply it to running directories to complete the upgrading of a schema in-place. It also shows how to view the version history of a directory schema, which helps to ensure the directory fleet is running the same version of the schema and has the correct history of schema changes applied to it.

Sample Schemas

Cloud Directory comes ready with sample schemas for Organizations, Persons, and Devices. The following section lists the various sample schemas and lists the differences for each.

Organizations

The following tables list the facets that are included in the *Organizations* sample schema.

"Organization" Facet	Data Type	Length	Require Behavi	Description
account_id	String	1024	N	Unique id for Organization
account_name	String	1024	N	Name of Organization
organization_status	String	1024	N	Status such as 'active', 'suspended', 'inactive', 'closed'
mailing_address (street1)	String	1024	N	A physical mailing address for this company/entity
mailing_address (street2)	String	1024	N	A physical mailing address for this company/entity

"Organization" Facet	Data Type	Length	Require Behavi	Description
mailing_address (city)	String	1024	N	A physical mailing address for this company/entity
mailing_address (state)	String	1024	N	A physical mailing address for this company/entity
mailing_address (country)	String	1024	N	A physical mailing address for this company/entity
mailing_address (postal_code)	String	1024	N	A physical mailing address for this company/entity
email	String	1024	N	Email id for Organization
web_site	String	1024	N	Website URL
telephone_number	String	1024	N	Telephone number for Organization
description	String	1024	N	Description for Organization

"Legal_Entity" Facet	Data Type	Length	Require Behavi	Description
registered_company_name	String	1024	N	Legal entity name
mailing_address (street1)	String	1024	N	A physical registered address for this company/entity
mailing_address (street2)	String	1024	N	A physical registered address for this company/entity
mailing_address (city)	String	1024	N	A physical registered address for this company/entity
mailing_address (state)	String	1024	N	A physical registered address for this company/entity
mailing_address (country)	String	1024	N	A physical registered address for this company/entity
mailing_address (postal_code)	String	1024	N	A physical registered address for this company/entity
industry_vertical	String	1024	N	Industry Segment
billing_currency	String	1024	N	Billing currency
tax_id	String	1024	N	Tax identification number

Person

The following tables list the facets that are included in the *Person* sample schema.

"Person" Facet	Data Type	Length	Required Behavior?	Description
display_name	String	1024	N	The name of the user, suitable for display to end-users.
first_name	String	1024	N	The given name of the User, or first name in most western languages
last_name	String	1024	N	The family name of the User, or last name in most western languages
middle_name	String	1024	N	The middle name(s) of the User
nickname	String	1024	N	The casual way to address the user in real life, such as, "Bob" or "Bobby" instead of "Robert"
email	String	1024	N	Email address for the user
mobile_phone_number	String	1024	N	Phone number for the user
home_phone_number	String	1024	N	Phone number for the user
username	String	1024	Y	unique identifier for the user
profile	String	1024	N	A URI that is a uniform resource locator and that points to a location representing the user's online profile (such as a webpage)
picture	String	1024	N	A URI that is a uniform resource locator that points to a resource location representing the user's image.
website	String	1024	N	URL
timezone	String	1024	N	The User's time zone
locale	String	1024	N	Used to indicate the User's default location for purposes of localizing such items as currency, date time format, or numerical representations.
address (street1)	String	1024	N	A physical mailing address for this user.
address (street2)	String	1024	N	A physical mailing address for this user.
address (city)	String	1024	N	A physical mailing address for this user.

"Person" Facet	Data Type	Length	Required Behavior?	Description
address (state)	String	1024	N	A physical mailing address for this user.
address (country)	String	1024	N	A physical mailing address for this user.
address (postal_code)	String	1024	N	A physical mailing address for this user.
user_status	String	1024	N	Value indicating the user's administrative status

"Organization_Person" Facet	Data Type	Length	Required Behavior?	Description
title	String	1024	N	Title in organization
preferred_language	String	1024	N	Indicates the user's preferred written or spoken languages and is generally used for selecting a localized user interface.
employee_id	String	1024	N	A string identifier, typically numeric or alphanumeric, assigned to a person
cost_center	Integer	1024	N	Identifies the cost center
department	String	1024	N	Identifies the name of a department
manager	String	1024	N	The user's manager
company_name	String	1024	N	Identifies the name of an organization
company_address (street1)	String	1024	N	A physical mailing address for the organization
company_address (street2)	String	1024	N	A physical mailing address for the organization
company_address (city)	String	1024	N	A physical mailing address for the organization
company_address (state)	String	1024	N	A physical mailing address for the organization
company_address (country)	String	1024	N	A physical mailing address for the organization
company_address (postalCode)	String	1024	N	A physical mailing address for the organization

Device

The following table lists the facet that is included in the *Device* sample schema.

"Device" Facet	Data Type	Length	Required Behavior?	Description
device_id	String	1024	N	Alpha-numeric unique device id
name	String	1024	N	Friendly name for device
description	String	1024	N	Description for device
X.509_certificates	String	1024	N	X.509 Certificate
device_version	String	1024	N	Device version
device_os_type	String	1024	N	Operating System on device
device_os_version	String	1024	N	Operating System version number on device
serial_number	String	1024	N	Serial number of device
device_status	String	1024	N	Status for device (such as active, not_active, suspended, shutdown, off)

Custom Schemas

The first step in creating a custom schema is to define exactly what fields you must index. These required fields form your schema's skeleton elements, to which you add your own fields. Map the name and type of each field (such as string, integer, Boolean) to your object's structure. You can define a schema with types and constraints and then apply them to a directory. Once defined, Cloud Directory performs validation for attributes.

For more information, see Create a Schema (p. 15).

Attribute Rules

Rules describe permissible values of an attribute type and constrain the values that are allowed for any particular attribute. You must specify rules as part of an attribute definition when you create a facet. Cloud Directory supports the following rule types:

- String length
- Binary length
- String from set
- Number comparison

String length

Constrains the length of a string attribute value.

Allowed rule parameter keys: min, max

Allowed rule parameter values: number

Binary length

Constrains the byte array length of a binary attribute value.

Allowed rule parameter keys: min, max

Allowed rule parameter values: number

String from set

Constrains the value of a string attribute to the allowed set of specified strings.

Allowed rule parameter keys: allowedValues

Allowed rule parameter values: Set of strings with each string to be UTF-8 encoded

Allowed values are comma delimited and can be wrapped in quotes. This is useful when allowed values include comma's. For example:

- One,two,three = matches One two or three
- "with,comma","withoutcomma" = matches "with,comma" or "withoutcomma"
- with"quote,withoutquote matches 'with"quote' or 'withoutquote'

Number comparison

Constraints the numeric value allowed for a number attribute.

Allowed rule parameter keys: min, max

Allowed rule parameter values: number

Format Specification

A Cloud Directory schema adds structure to the data in your data directories. Cloud Directory provides two mechanisms for you to define your schema. Developers can use specific API operations to construct a schema or they can upload a schema entirely using schema upload capabilities. Schema documents can be uploaded via API calls or through the console. This section describes the format to use when you upload entire schema documents.

JSON Schema Format

A schema document is a JSON document in the following overall format.

```
{
     “facets”: {
            “facet name”: {
  “facetAttributes”: {
   “attribute name”: Attribute JSON Subsection
                }
        }
     }
}
```

A schema document contains a map of facet names to facets. Each facet in turn contains a map containing attributes. All facet names within a schema must be unique. All attribute names within a facet must be unique.

Attribute JSON Subsection

Facets contain attributes. Each attribute defines the type of value that can be stored on an attribute. The following JSON format describes an attribute.

```
{
     “attributeDefinition”: Attribute Definition Subsection,
     “attributeReference”: Attribute Reference Subsection,
     “requiredBehavior”: “REQUIRED_ALWAYS or NOT_REQUIRED”
}
```

You must provide either an attribute definition or an attribute reference. See related subsections for more information about each.

The required behavior field indicates whether this attribute is required or not. You must provide this field. Possible values are as follows:

- `REQUIRED_ALWAYS`: This attribute must be provided when the object is created or a facet is added to the object. You cannot remove this attribute.
- `NOT_REQUIRED`: This attribute may or may not be present.

Attribute Definition Subsection

An attribute defines the type and the rules associated with an attribute value. The following JSON layout describes the format.

```
{
 “attributeType”: “One of STRING, NUMBER, BINARY, BOOLEAN, DATETIME”
 “defaultValue”: Default Value Subsection
      “isImmutable”: true or false
 “attributeRules”: Attribute Rules Subsection
}
```

Default Value Subsection

Specify exactly one of the following default values. Long values and Boolean values should be provided outside of quotes (as their respective Javascript types instead of strings). Binary values are provided using a URL-safe Base64 encoded string (as described in RFC 4648). Datetimes are provided in the number of milliseconds since the epoch (00:00:00 UTC on Jan 1, 1970).

```
{
 “stringValue”: “a string value”,
 “longValue”: an integer value,
 “booleanValue”: a boolean value,
 “binaryValue”: a URL-safe Base64 encoded string,
 “datetimeValue”: an integer value representing milliseconds since epoch
}
```

Attribute Rules Subsection

Attributes rules define constraints on attribute values. You can define multiple rules for each attribute. Attribute rules contain a rule type and a set of parameters for the rule. You can find more details in the Attribute Rules (p. 35) section.

```
{
 “rule name”: {
  “parameters”: {
   “rule parameter key 1”: “value”,
   “rule parameter key 2”: “value”
  },
  “ruleType”: “rule type value”
 }
}
```

Attribute Reference Subsection

Attribute references are an advanced feature. They allow multiple facets to share an attribute definition and stored value. See the Attribute References (p. 50) section for more information. You can define an attribute reference in JSON schema with the following template.

```
{
 “targetSchemaArn”: “schema ARN”
 “targetFacetName”: “facet name”
 “targetAttributeName”: “attribute name”
}
```

Schema Document Examples

The following are samples of schema documents that show valid JSON formatting.

> **Note**
> All values expressed in the `allowedValues` string must be comma separated and be without spaces. For example, `"SENSITIVE,CONFIDENTIAL,PUBLIC"`.

Basic Schema Document

```
{
  "facets" : {
    "Employee" : {
      "facetAttributes" : {
        "Name" : {
          "attributeDefinition" : {
            "attributeType" : "STRING",
            "isImmutable" : false,
            "attributeRules" : {
              "NameLengthRule" : {
                "parameters" : {
                  "min" : "3",
                  "max" : "100"
                },
                "ruleType": "STRING_LENGTH"
              }
            }
          },
          "requiredBehavior" : "REQUIRED_ALWAYS"
        },
        "EmailAddress" : {
          "attributeDefinition" : {
            "attributeType" : "STRING",
            "isImmutable" : true,
            "attributeRules" : {
              "EmailAddressLengthRule" : {
                "parameters" : {
                  "min" : "3",
                  "max" : "100"
                },
                "ruleType": "STRING_LENGTH"
              }
            }
          },
          "requiredBehavior" : "REQUIRED_ALWAYS"
        },
        "Status" : {
          "attributeDefinition" : {
            "attributeType" : "STRING",
            "isImmutable" : false,
            "attributeRules" : {
              "rule1" : {
                "parameters" : {
                  "allowedValues" : "ACTIVE,INACTIVE,TERMINATED"
                },
                "ruleType": "STRING_FROM_SET"
              }
            }
          },
          "requiredBehavior" : "REQUIRED_ALWAYS"
        }
      },
      "objectType" : "LEAF_NODE"
    },
    "DataAccessPolicy" : {
      "facetAttributes" : {
        "AccessLevel" : {
          "attributeDefinition" : {
            "attributeType" : "STRING",
            "isImmutable" : true,
            "attributeRules" : {
              "rule1" : {
                "parameters" : {
                  "allowedValues" : "SENSITIVE,CONFIDENTIAL,PUBLIC"
                },
                "ruleType": "STRING_FROM_SET"
              }
            }
          },
          "requiredBehavior" : "REQUIRED_ALWAYS"
        }
      },
      "objectType" : "POLICY"
    },
    "Group" : {
      "facetAttributes" : {
        "Name" : {
          "attributeDefinition" : {
            "attributeType" : "STRING",
            "isImmutable" : true
          },
          "requiredBehavior" : "REQUIRED_ALWAYS"
        }
      },
      "objectType" : "NODE"
    }
  }
}
```

Schema Document with Typed Links

```
{
 "sourceSchemaArn": "",
 "facets": {
  "employee_facet": {
   "facetAttributes": {
    "employee_login": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    },
    "employee_id": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    },
    "employee_name": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    },
    "employee_role": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    }
   },
   "objectType": "LEAF_NODE"
  },
  "device_facet": {
   "facetAttributes": {
    "device_id": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    },
    "device_type": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    }
   },
   "objectType": "NODE"
  },
  "region_facet": {
   "facetAttributes": {},
   "objectType": "NODE"
  },
  "group_facet": {
   "facetAttributes": {
    "group_type": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    }
   },
   "objectType": "NODE"
  },
  "office_facet": {
   "facetAttributes": {
    "office_id": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    },
    "office_type": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    },
    "office_location": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": true,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    }
   },
   "objectType": "NODE"
  }
 },
 "typedLinkFacets": {
  "device_association": {
   "facetAttributes": {
    "device_type": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": false,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    },
    "device_label": {
     "attributeDefinition": {
      "attributeType": "STRING",
      "isImmutable": false,
      "attributeRules": {}
     },
     "requiredBehavior": "REQUIRED_ALWAYS"
    }
   },
   "identityAttributeOrder": ["device_label", "device_type"]
  }
 }
}
```

Indexing and Search

Amazon Cloud Directory supports two methods of indexing: Value based and type based. Value-based indexing is the most common form. With it you can index and search for objects in the directory based on the values of object attributes. With type-based indexing, you can index and search for objects in the directory based on object types. Facets help define object types. For more information about schemas and facets, see Schemas (p. 27) and Facets (p. 30).

Indexes in Cloud Directory enable simple listing of other objects by those objects' attribute or facet values. Each index is defined at creation to work with a specific named attribute or facet. For example, an index may be defined on the “email” attribute of the “Person” facet. Indexes are first-class objects, which means that clients can create, modify, list, and delete them flexibly according to the application logic’s needs.

Conceptually, indexes are similar to nodes with children, where the links to the indexed nodes are labeled according to the indexed attributes, rather than being given a label when the child is attached. However, index links are not parent-child edges, and have their own set of enumeration API operations.

It’s important to understand that indexes in Cloud Directory are not automatically populated as they might be in other systems. Instead, you use API calls to directly attach and detach objects to or from the index. Although this is a bit more work, it gives you flexibility to define varying index scopes. For example, you can define an index that tracks only direct children of a specific node. Or you can define an index that tracks all objects in a given branch under a local root, such as all nodes in a department. You can also do both at the same time.

Index lifecycle

You can use the following API calls to help with the development lifecycle of indexes.

1. You create indexes with the `CreateIndex` API call. You supply an index definition structure that describes the attributes on attached objects that the index will track. The definition also indicates whether or not the index should enforce uniqueness. The result is an object ID for the new index, which should immediately be attached to your hierarchy like any other object. For example, this can be a branch dedicated to holding indexes.
2. You attach objects to the index manually with the `AttachToIndex` API call. The index then automatically tracks the values of its defined attributes on each attached object.
3. To use the indexes to search for objects with more efficient enumeration, call `ListIndex` and specify a range of values that you are interested in.
4. Use the `ListAttachedIndices` API call to enumerate the indexes that are attached to a given object.
5. Use the `DetachFromIndex` API call to remove objects from the index manually.
6. Once you detach all objects from the index, you can delete the index with the `DeleteObject` API call.

There is no limit on the number of indexes within a directory, other than the limit on the space used by all objects. Indexes and their attachments do consume space, but it is similar to that consumed by nodes and parent–child links. There is a limit on the number of indexes that can be attached to a given object. For more information, see AWS Directory Service Limits (p. 194).

Facet-Based Indexing

With facet-based indexing and search, you can optimize your directory searches by searching only a subset of your directory. To do this, you use a schema *facet*. For example, instead of searching across all the user objects in your directory, you can search only the user objects that contain an employee facet. This efficiency helps reduce the latency time and amount of data retrieved for the query.

With facet-based indexing, you can use the Cloud Directory index API operations to create and attach an index to the facets of objects. You can also list the index results and then filter those results based on certain facets. This can effectively reduce query times and the amount of data by narrowing the search scope to only objects that contain a certain type of facet.

The "`facets`" attribute that is used with the `CreateIndex` and `ListIndex` API calls surfaces the collection of facets that are applied to an object. This attribute is available for use only with the `CreateIndex` and `ListIndex` API calls. As shown in the following sample code, the schema ARN uses the directory's region, owner account, and directory ID to reference the Cloud Directory schema. This service-provided schema does not appear in listings.

```
String cloudDirectorySchemaArn = String.format("arn:aws:clouddirectory:%s:%s:directory/%s/
schema/CloudDirectory/1.0", region, ownerAccount, directoryId);
```

For example, the following sample code creates a facet-based index specific to your AWS account and directory where you could enumerate all objects that are created with the facet `SalesDepartmentFacet`.

Note
Make sure to use the "facets" value within the parameters as shown below. Instances of "facets" shown in the sample code refer to a value that is provided and controlled by the Cloud Directory service. You can use these for indexing but can have read-only access.

```
// Create a facet-based index
String cloudDirectorySchemaArn = String.format("arn:aws:clouddirectory:%s:%s:directory/%s/
schema/CloudDirectory/1.0",
    region, ownerAccount, directoryId);

facetIndexResult = clouddirectoryClient.createIndex(new CreateIndexRequest()
  .withDirectoryArn(directoryArn)
  .withOrderedIndexedAttributeList(List(new AttributeKey()
        .withSchemaArn(cloudDirectorySchemaArn)
        .withFacetName("facets")
        .withName("facets")))
        .withIsUnique(false)
        .withParentReference("/")
        .withLinkName("MyFirstFacetIndex"))
facetIndex = facetIndexResult.getObjectIdentifier()

// Attach objects to the facet-based index
clouddirectoryClient.attachToIndex(new
 AttachToIndexRequest().withDirectoryArn(directoryArn)
  .withIndexReference(facetIndex).withTargetReference(userObj))

// List all objects
val listResults = clouddirectoryClient.listIndex(new ListIndexRequest()
  .withDirectoryArn(directoryArn)
  .withIndexReference(facetIndex)
  .getIndexAttachments()

// List the index results filtering for a certain facet
val filteredResults = clouddirectoryClient.listIndex(new ListIndexRequest()
  .withDirectoryArn(directoryArn)
  .withIndexReference(facetIndex)
  .withRangesOnIndexedValues(new ObjectAttributeRange()
    .withAttributeKey(new AttributeKey()
      .withFacetName("facets")
      .withName("facets")
      .withSchemaArn(cloudDirectorySchemaArn))
    .withRange(new TypedAttributeValueRange()
      .withStartMode(RangeMode.INCLUSIVE)
      .withStartValue("MySchema/1.0/SalesDepartmentFacet")
      .withEndMode(RangeMode.INCLUSIVE)
      .withEndValue("MySchema/1.0/SalesDepartmentFacet")
    )))
```

Unique vs Nonunique Indexes

Unique indexes differ from nonunique indexes in enforcing uniqueness of the indexed attribute values for objects that are attached to the index. For example, you might want to populate Person objects into two indexes, a unique one on an “email” attribute, and a nonunique one on a “lastname” attribute. The lastname index allows many Person objects with the same last name to be attached. On the other hand, the `AttachToIndex` call that targets the email index returns a `LinkNameAlreadyInUseException` error if a Person with the same email attribute is already attached. Note that the error does not remove the Person object itself. Consequently, an application might create the Person, attach it to the hierarchy, and attach it to indexes, all in a single batch request. This ensures that if uniqueness is violated on any of the indexes, the object and all of its attachments are rolled back automatically.

Using the Cloud Directory APIs

Amazon Cloud Directory includes a set of API operations that enable programmatic access to Cloud Directory capabilities. You can use the Amazon Cloud Directory API Reference Guide to learn how to make requests to the Cloud Directory API for creating and managing the various elements. It also covers the components of requests, the content of responses, and how to authenticate requests.

Cloud Directory provides all necessary API operations that enable developers to build new applications. It provides the following categories of API calls:

- Create, read, update, delete (CRUD) for schema
- CRUD for facet
- CRUD for directories
- CRUD for objects (nodes, policies, etc.)
- CRUD for Index definition
- Batch read, batch write

How Billing Works With Cloud Directory APIs

Billing for API calls vary based on the specific types of API calls being made. There are specific billing rates for Eventually Consistent Read API calls, Strongly Consistent Read API calls, and Write API calls. Metadata API calls are free.

Strongly Consistent operations are used for read-after-write consistency when reading a value. Eventually Consistent operations are used for retrieving a value while updates are running. With Eventually Consistent operations retrieved results might not be the most accurate since the specific host you are reading the value from is still processing updates. However, the latency for such read operations are low when you retrieve a performance call.

When reading data from Cloud Directory, you must specify either an Eventually Consistent Read or Strongly Consistent Read type operation. The read type is based on consistency level. The two consistency levels are EVENTUAL for Eventually Consistent Reads and SERIALIZABLE for Strongly Consistent Reads. For more information, see Consistency Levels (p. 26).

The following table lists all of the Cloud Directory APIs and how they can impact billing for your AWS account.

API	Eventually Consistent Read [1]	Strongly Consistent Read [2]	Write [3]	Metadata [4]
AddFacetToObject			X	
ApplySchema				X
AttachObject			X	
AttachPolicy			X	
AttachToIndex			X	
AttachTypedLink			X	
BatchRead	X	X		
BatchWrite			X	
CreateDirectory			X	
CreateFacet				X
CreateIndex			X	
CreateObject			X	
CreateSchema				X
CreateTypedLinkFacet				X
DeleteDirectory				X
DeleteFacet				X
DeleteObject			X	
DeleteSchema				X
DetachFromIndex			X	
DetachObject			X	
DetachPolicy			X	
DetachTypedLink			X	
DeleteTypedLinkFacet				X
DisableDirectory				X
EnableDirectory			X	
GetAppliedSchemaVersion				X
GetDirectory				X
GetFacet				X
GetObjectInformation	X	X		
GetSchemaAsJson				X

API	Eventually Consistent Read [1]	Strongly Consistent Read [2]	Write [3]	Metadata [4]
GetTypedLinkFacetInformation				X
ListAppliedSchemaArns				X
ListAttachedIndices	X	X		
ListDevelopmentSchemaArns				X
ListDirectories				X
ListFacetAttributes				X
ListFacetNames				X
ListIncomingTypedLinks	X	X		
ListIndex	X	X		
ListObjectAttributes	X	X		
ListObjectChildren	X	X		
ListObjectParentPaths	X			
ListObjectParents	X	X		
ListObjectPolicies	X	X		
ListOutgoingTypedLinks	X	X		
ListPolicyAttachments	X	X		
ListPublishedSchemaArns				X
ListTagsForResource				X
ListTypedLinkFacetAttributes				X
ListTypedLinkFacetNames				X
LookupPolicy	X			
PublishSchema				X
PutSchemaFromJson				X
RemoveFacetFromObject			X	
TagResource				X
UntagResource				X
UpdateFacet				X
UpdateObjectAttributes			X	
UpdateSchema				X
UpdateTypedLinkFacet				X

API	Eventually Consistent Read [1]	Strongly Consistent Read [2]	Write [3]	Metadata [4]
UpgradeAppliedSchema				X
UpgradePublishedSchema				X

[1] Eventually Consistent Read APIs are called with the EVENTUAL consistency level

[2] Strongly Consistent Read APIs are called with the SERIALIZABLE consistency level

[3] Write APIs are billed as write API calls

[4] Metadata APIs are NOT billed but are categorized as Metadata API calls

For additional information about billing, see Amazon Cloud Directory Pricing.

Amazon Cloud Directory Compliance

Amazon Cloud Directory has undergone auditing for the following standards and is eligible for use as part of solutions for which you need to obtain compliance certification.

	Cloud Directory has an Attestation of Compliance for Payment Card Industry (PCI) Data Security Standard (DSS) version 3.2 at Service Provider Level 1. Customers who use AWS products and services to store, process, or transmit cardholder data can use Cloud Directory as they manage their own PCI DSS compliance certification. For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see PCI DSS Level 1.
	AWS has expanded its Health Insurance Portability and Accountability Act (HIPAA) compliance program to include Amazon Cloud Directory as a HIPAA Eligible Service. If you have an executed Business Associate Agreement (BAA) with AWS, you can use Cloud Directory to help build your HIPAA-compliant applications. AWS offers a HIPAA-focused Whitepaper for customers who are interested in learning more about how they can leverage AWS for the processing and storage of health information. For more information, see HIPAA Compliance

Shared Responsibility

Security, including HIPAA and PCI compliance, is a shared responsibility. It is important to understand that Cloud Directory compliance status does not automatically apply to applications that you run in the AWS Cloud. You need to ensure that your use of AWS services complies with the standards.

Advanced Features

Amazon Cloud Directory includes a set of advanced features that can be helpful when you want to customize your applications.

Topics

Batches

With Amazon Cloud Directory, it’s often necessary to add new objects or add relationships between new objects and existing objects to reflect changes in a real-world hierarchy. Batch operations can make directory tasks like these easier to manage by providing the following benefits:

- Batch operations can minimize the number of round trips required to write and read objects to and from your directory, improving the overall performance of your application.
- Batch write provides the SQL database-equivalent transaction semantics. All operations successfully complete, or if any operation has a failure then none of them are applied.
- Using batch reference you can create an object and use a reference to the new object for further action such as adding it to a relationship, reducing overhead of using a read operation before a write operation.

BatchWrite

Use BatchWrite operations to perform multiple write operations on a directory. All operations in batch write are executed sequentially. It works similar to SQL database transactions. If one of the operation inside batch write fails, the entire batch write has no effect on the directory. If a batch write fails, a batch write exception occurs. The exception contains the index of the operation that failed along with exception type and message. This information can help you identify the root cause for the failure.

The following API operations are supported as part of batch write:

- AddFacetToObject
- AttachObject
- AttachPolicy
- AttachToIndex
- AttachTypedLink
- CreateIndex
- CreateObject
- DeleteObject
- DetachFromIndex
- DetachObject
- DetachTypedLink
- RemoveFacetFromObject
- UpdateObjectAttributes

Batch Reference Name

Batch reference names are supported only for batch writes when you need to refer to an object as part of the intermediate batch operation. For example, suppose that as part of a given batch write, 10 different objects are being detached and are attached to a different part of the directory. Without batch reference, you would have to read all 10 object references and provide them as input during reattachment as part of the batch write. You can use a batch reference to identify the detached resource during attachment. A batch reference can be any regular string prefixed with the number sign / hashtag symbol (#).

For example, in the following code sample, an object with link name `"this-is-a-typo"` is being detached from root with a batch reference name `"ref"` . Later the same object is attached to the root with the link name as `"correct-link-name"` . The object is identified with the child reference set to batch reference. Without the batch reference, you would initially need to get the `objectIdentifier` that is being detached and provide that in the child reference during attachment. You can use a batch reference name to avoid this extra read.

```
BatchDetachObject batchDetach = new BatchDetachObject()
        .withBatchReferenceName("ref")
        .withLinkName("this-is-a-typo")
        .withParentReference(new ObjectReference().withSelector("/"));
BatchAttachObject batchAttach = new BatchAttachObject()
        .withParentReference(new ObjectReference().withSelector("/"))
        .withChildReference(new ObjectReference().withSelector("#ref"))
        .withLinkName("correct-link-name");
BatchWriteRequest batchWrite = new BatchWriteRequest()
        .withDirectoryArn(directoryArn)
        .withOperations(new ArrayList(Arrays.asList(batchDetach, batchAttach)));
```

BatchRead

Use BatchRead operations to perform multiple read operations on a directory. For example, in the following code sample, children of object with reference `“/managers”` is being read along with attributes of object with reference `“/managers/bob”` in a single batch read.

```
BatchListObjectChildren listObjectChildrenRequest = new BatchListObjectChildren()
        .withObjectReference(new ObjectReference().withSelector("/managers"));
BatchListObjectAttributes listObjectAttributesRequest = new BatchListObjectAttributes()
        .withObjectReference(new ObjectReference().withSelector("/managers/bob"));
BatchReadRequest batchRead = new BatchReadRequest()
        .withConsistencyLevel(ConsistencyLevel.SERIALIZABLE)
        .withDirectoryArn(directoryArn)
        .withOperations(new ArrayList(Arrays.asList(listObjectChildrenRequest,
 listObjectAttributesRequest)));
BatchReadResult result = cloudDirectoryClient.batchRead(batchRead);
```

BatchRead supports the following API operations:

- GetObjectInformation
- ListAttachedIndices
- ListIncomingTypedLinks
- ListIndex
- ListObjectAttributes
- ListObjectChildren
- ListObjectParentPaths
- ListObjectPolicies
- ListOutgoingTypedLinks

- ListPolicyAttachments
- LookupPolicy

Limits on Batch operations

Each request to the server (including batched requests) has a maximum number of resources that can be operated on, regardless of the number of operations in the request. This allows you to compose batch requests with high flexibility as long as you stay within the resource maximums. For more information on resource maximums, see AWS Directory Service Limits (p. 194).

Limits are calculated by summing the writes or reads for each single operation inside the Batch. For example, the read operation limit is currently 200 objects per API call. Let's say you want to compose a batch that adds 9 ListObjectChildren API calls and each call requires reading 20 objects . Since the total number of read objects (9 x 20 = 180) does not exceed 200, the batch operation would succeed.

The same concept applies with calculating write operations. For example, the write operation limit is currently 20. If you set up your batch to add 2 UpdateObjectAttributes API calls with 9 write operations each, this would also succeed. In either case, should the batch operation exceed the limit, then the operation will fail and a `LimitExceededException` will be thrown.

Exception handling

Batch operations in Cloud Directory can sometimes fail. In these cases, it is important to know how to handle such failures. The method you use to resolve failures differs for write operations and read operations.

Batch write operation failures

If a batch write operation fails, Cloud Directory fails the entire batch operation and returns an exception. The exception contains the index of the operation that failed along with the exception type and message. If you see `RetryableConflictException`, you can try again with exponential backoff. A simple way to do this is to double the amount of time you wait each time you get an exception or failure. For example, if your first batch write operation fails, wait 100 milliseconds and try the request again. If the second request fails, wait 200 milliseconds and try again. If the third request fails, wait 400 milliseconds and try again.

Batch read operation failures

If a batch read operation fails, the response contains either a successful response or an exception response. Individual batch read operation failures do not cause the entire batch read operation to fail—Cloud Directory returns individual success or failure responses for each operation.

Related Cloud Directory Blog Articles

- Write and Read Multiple Objects in Amazon Cloud Directory by Using Batch Operations
- How to Use Batch References in Amazon Cloud Directory to Refer to New Objects in a Batch Request

Attribute References

Amazon Cloud Directory facets contain attributes. Attributes can be either an attribute definition or an attribute reference. Attribute definitions are attributes that declare their name and primitive type (string, binary, Boolean, DateTime, or number). They can also optionally declare their required behavior, default value, immutable flag, and attribute rules (such as min/max length).

Attribute references are attributes that derive their primitive type, default value, immutable flag and attribute rules from another preexisting attribute definition. Attribute references do not have their own primitive type, default values, immutable flag or rules, since those properties come from the target attribute definition.

Attribute references may override the required behavior of a target definition (more details about this below).

When you create an attribute reference, you provide only an attribute name and the target attribute definition (which includes the facet name and attribute name of the target attribute definition). Attribute references may not refer to other attribute references. Also, at this time, attribute references may not target attribute definitions from a different schema.

You can use an attribute reference when you want two or more attributes on an object to refer to the same storage location. For example, imagine an object that has a User facet and an EnterpriseUser facet applied. The User facet has a FirstName attribute definition, while the EnterpriseUser facet has an attribute reference pointing at User.FirstName. Since both FirstName attributes refer to the same storage location on the object, any change to either the User.FirstName or the EnterpriseUser.FirstName has the same effect.

API Example

The following example demonstrates the use of attribute references using the Cloud Directory API. In this example, a base facet contains an attribute definition, and another facet contains an attribute referring to an attribute in the base facet. Note that the reference attribute can be marked Required while the base facet is Not Required.

```
   // create base facet
   CreateFacetRequest req1 = new CreateFacetRequest()
     .withSchemaArn(devSchemaArn)
     .withName("baseFacet")
     .withAttributes(List(
       new FacetAttribute()
         .withName("baseAttr")
         .withRequiredBehavior(RequiredAttributeBehavior.NOT_REQUIRED)
         .withAttributeDefinition(new
FacetAttributeDefinition().withType(FacetAttributeType.STRING))))
     .withObjectType(ObjectType.DIRECTORY)
   cloudDirectoryClient.createFacet(req1)

   // create another facet that refers to the base facet
   CreateFacetRequest req2 = new CreateFacetRequest()
     .withSchemaArn(devSchemaArn)
     .withName("facetA")
     .withAttributes(List(
       new FacetAttribute()
         .withName("ref")
         .withRequiredBehavior(RequiredAttributeBehavior.REQUIRED_ALWAYS)
         .withAttributeReference(new FacetAttributeReference()
           .withTargetFacetName("baseFacet")
           .withTargetAttributeName("baseAttr"))))
     .withObjectType(ObjectType.DIRECTORY)
   cloudDirectoryClient.createFacet(req2)
```

JSON Example:

The following example demonstrates the use of attribute references in a JSON model. The schema represented by this model is identical to the model above.

```
{
 "facets" : {
   "baseFacet" : {
      "facetAttributes" : {
        "baseAttr" : {
          "attributeDefinition" : {
            "attributeType" : "STRING"
          },
          "requiredBehavior" : "NOT_REQUIRED"
        }
      },
      "objectType" : "DIRECTORY"
    },
    "facetA" : {
      "facetAttributes" : {
        "ref" : {
          "attributeReference" : {
            "targetFacetName" : "baseFacet",
            "targetAttributeName" : "baseAttr"
          },
          "requiredBehavior" : "REQUIRED_ALWAYS"
        }
      },
      " objectType" : "DIRECTORY"
    }
}
```

Attribute reference considerations

Attribute references must target a preexisting attribute definition in the same schema.

- Attribute references may target a preexisting attribute definition in the same facet or a different facet.
- Attribute references may not target other attribute references.
- Facets containing attribute definitions that are the target of another facet's attribute reference cannot be deleted until all references have been deleted.

You can use attribute references the same way that you use traditional attribute definitions, by creating objects or applying facets to existing objects.

> **Note**
> You can apply facets with references to other facets but are not required to apply the target facets directly. When the target facet is not applied, there is no change to the behavior of the attribute reference. (You need to apply target facets only when you want the other attributes on that facet to exist on the object.)

Setting Attribute Reference Values

You can call the UpdateObjectAttributes API action when you want to change the value of an attribute. Updating (or deleting) either the definition or any other reference to that same definition on that object has the same effect.

Getting Attribute Reference Values

You can call the ListObjectAttributes API action to retrieve storage aliases. This call returns a list of tuples, each of which contains an attribute key and its associated value. The attribute keys correspond to the list of storage aliases present on that object.

> **Note**
> It is possible for an attribute key to be returned for a facet that was not explicitly applied to an object. This can happen when attribute references target facets that are not applied to the object.

For example, imagine that you have a User facet and an EnterpriseUser facet. The EnterpriseUser.FirstName attribute refers to User.FirstName. You then apply both the User and

EnterpriseUser facets to an object, set User.FirstName to Robert, and later set EnterpriseUser.FirstName to Bob. When you call ListObjectAttributes you see only "User.FirstName = Bob" since there is only one storage alias for both FirstName attributes.

Using Indexes with Attribute References

You can can create indexes with an attribute definition only, not a reference. Listing an index does not return attribute keys for attribute references. But it does return attribute keys for any attribute definitions that are targeted by references existing on the indexed object. In other words, at the index layer, attribute references are treated merely as an alternative identifier for an attribute, which gets resolved to the correct attribute definition identifier at runtime.

For example, imagine you have an Index for facet User attribute FirstName. You attach an object with only the EnterpriseUser facet applied. You then set the value for that object's EnterpriseUser.FirstName attribute to Bob. Finally, you call ListIndex action. The results contain only "User.FirstName = Bob".

Required Behavior for Attribute References

An attribute references can have a required behavior that is different from its target attribute definition. This allows a base definition to be optional, while a reference to that same definition can be required. When an object has a base definition and one or more references to the same base definition, the base definition and all references must adhere to the strongest required behavior present across all related attributes.

- As with attribute definitions, you must provide values for any required attribute definitions when you create the object or when you add a facet to an existing object.
- As a convenience, when more than one attribute on an object refers to the same storage location, you only need to provide a value for one of the attributes for that storage location.
- Similarly, if you do provide multiple values for the same storage location, the values must be equal.

Cloud Directory Resources

The following tables list related resources that you'll find useful as you work with this service.

Cloud Directory Blog Posts	Description
Write and Read Multiple Objects in Amazon Cloud Directory by Using Batch Operations	Explains about using Batch Read and Write. It also includes sample Java code.
How to Use Batch References in Amazon Cloud Directory to Refer to New Objects in a Batch Request	Explains about using Batch Reference. It also includes sample Java code.
Cloud Directory Update – Support for Typed Links	Explains about creating and searching relationships across hierarchies in Cloud Directory by using typed links. It also includes sample Java code.
New Cloud Directory API Makes It Easier to Query Data Along Multiple Dimensions	Explains how to query for data across multiple dimensions with a single call using the `ListObjectParentPaths` API.
How to Create an Organizational Chart with Separate Hierarchies by Using Amazon Cloud Directory	Explains how to create a schema and a directory with sample Java code.

Cloud Directory Blog Posts	Description
Amazon Cloud Directory – A Cloud-Native Directory for Hierarchical Data	Describes the launch of Cloud Directory as a new service from AWS.

Cloud Directory Documentation	Link
Cloud Directory Developer Guide	http://docs.aws.amazon.com/directoryservice/latest/admin-guide/directory_amazon_cd.html
Cloud Directory API Reference	http://docs.aws.amazon.com/amazoncds/latest/APIReference/welcome.html
Cloud Directory Limits	http://docs.aws.amazon.com/directoryservice/latest/admin-guide/limits.html#limits_cd

Cloud Directory Other	Link
Cloud Directory Webinar	https://www.youtube.com/watch?v=UANm3DC_IxE
Cloud Directory Sample Java Code	https://github.com/aws-samples/AmazonCloudDirectory-sample
Cloud Directory Product Info	https://aws.amazon.com/cloud-directory/
Cloud Directory Pricing	https://aws.amazon.com/cloud-directory/pricing/

Microsoft Active Directory

AWS Directory Service lets you run Microsoft Active Directory (AD) as a managed service. AWS Directory Service for Microsoft Active Directory, also referred to as Microsoft AD, is powered by Windows Server 2012 R2. When you select and launch this directory type, it is created as a highly available pair of domain controllers connected to your virtual private cloud (VPC). The domain controllers run in different Availability Zones in a region of your choice. Host monitoring and recovery, data replication, snapshots, and software updates are automatically configured and managed for you.

With Microsoft AD, you can run directory-aware workloads in the AWS Cloud, including Microsoft SharePoint and custom .NET and SQL Server-based applications. You can also configure a trust relationship between Microsoft AD in the AWS Cloud and your existing on-premises Microsoft Active Directory, providing users and groups with access to resources in either domain, using single sign-on (SSO).

AWS Directory Service makes it easy to set up and run directories in the AWS Cloud, or connect your AWS resources with an existing on-premises Microsoft Active Directory. Once your directory is created, you can use it for a variety of tasks:

- Manage users and groups
- Provide single sign-on to applications and services
- Create and apply group policy
- Securely connect to Amazon EC2 Linux and Windows instances
- Simplify the deployment and management of cloud-based Linux and Microsoft Windows workloads
- You can use Microsoft AD to enable multi-factor authentication by integrating with your existing RADIUS-based MFA infrastructure to provide an additional layer of security when users access AWS applications.

Read the topics in this section to get started creating a Microsoft AD directory, creating a trust relationship between Microsoft AD and your on-premises directories, and extending your Microsoft AD schema.

Topics

Related AWS Security Blog Articles

- How to Configure Even Stronger Password Policies to Help Meet Your Security Standards by Using AWS Directory Service for Microsoft AD

- How to Increase the Redundancy and Performance of Your AWS Directory Service for Microsoft AD by Adding Domain Controllers
- How to Enable the Use of Remote Desktops by Deploying Microsoft Remote Desktop Licensing Manager on Microsoft AD
- How to Access the AWS Management Console Using Microsoft AD and Your On-Premises Credentials
- How to Enable Multi-Factor Authentication for AWS Services by Using Microsoft AD and On-Premises Credentials
- How to Easily Log On to AWS Services by Using Your On-Premises Active Directory

Create a Microsoft AD Directory

Microsoft AD creates a fully managed, Microsoft Active Directory in the AWS cloud and is powered by Windows Server 2012 R2 and operates at the 2012 R2 functional level. When you create a directory with Microsoft AD, AWS Directory Service creates two domain controllers and adds the DNS service on your behalf. The domain controllers are created in different subnets in a VPC; this redundancy helps ensure that your directory remains accessible even if a failure occurs. If you need more domain controllers, you can add them later. For more information, see Deploy Additional Domain Controllers (p. 114).

Topics

Microsoft AD Prerequisites

To create a Microsoft AD directory, you need a VPC with the following:

- At least two subnets. Each of the subnets must be in a different Availability Zone.
- The following ports must be open between the two subnets that you deploy your directory into. This is necessary to allow the domain controllers that AWS Directory Service creates for you to communicate with each other.
  - TCP/UDP 53 - DNS
  - TCP/UDP 88 - Kerberos authentication
  - UDP 123 - NTP
  - TCP 135 - RPC
  - UDP 137-138 - Netlogon
  - TCP 139 - Netlogon
  - TCP/UDP 389 - LDAP
  - TCP/UDP 445 - SMB
  - TCP 873 - Rsync
  - TCP 3268 - Global Catalog
  - TCP/UDP 1024-65535 - Ephemeral ports for RPC
- The VPC must have default hardware tenancy.
- You cannot create a Microsoft AD in a VPC using addresses in the 198.19.0.0/16 address space.
- AWS Directory Service does not support using Network Address Translation (NAT) with Active Directory. Using NAT can result in replication errors.

Multi-factor Authentication Prerequisites

To support multi-factor authentication with your Microsoft AD directory, you must configure your on-premises Remote Authentication Dial-In User Service (RADIUS) server in the following way so that it can accept requests from your Microsoft AD directory in AWS.

1. On your RADIUS server, create two RADIUS clients to represent both of the Microsoft AD domain controllers (DCs) in AWS. You will need to configure both clients using the following common parameters (your RADIUS server may vary):
   - **Address (DNS or IP)**: This is the DNS address for one of the Microsoft AD DCs. Both DNS addresses can be found in the AWS Directory Service Console on the **Details** page of the Microsoft AD directory in which you plan to use MFA. The DNS addresses displayed represent the IP addresses for both of the Microsoft AD DCs that are used by AWS.

     Note
     If your RADIUS server supports DNS addresses, you will need to create only one RADIUS client configuration. Otherwise, you must create one RADIUS client configuration for each Microsoft AD DC.
   - **Port number**: Configure the port number for which your RADIUS server accepts RADIUS client connections. The standard RADIUS port is 1812.
   - **Shared secret**: Type or generate a shared secret that will be used by the RADIUS server to connect with RADIUS clients.
   - **Protocol**: You might need to configure the authentication protocol between the Microsoft AD DCs and the RADIUS server. Supported protocols are PAP, CHAP MS-CHAPv1, and MS-CHAPv2. MS-CHAPv2 is recommended because it provides the strongest security of the three options.
   - **Application name**: This may be optional in some RADIUS servers and usually identifies the application in messages or reports.
2. Configure your on-premises network to allow inbound traffic from the RADIUS clients (Microsoft AD DCs DNS addresses, see Step 1) to your RADIUS server port.
3. Add a rule to the Amazon EC2 security group in your Microsoft AD domain that allows inbound traffic from the RADIUS server DNS address and port number defined previously. For more information, see Adding Rules to a Security Group in the *EC2 User Guide*.

For more information about using Microsoft AD with MFA, see Multi-Factor Authentication (p. 109).

How to Create a Microsoft AD directory

To create a new directory, perform the following steps. Before starting this procedure, make sure you have completed the prerequisites identified in Microsoft AD Prerequisites (p. 56).

To create a Microsoft AD directory

1. In the AWS Directory Service console navigation pane, select **Directories** and choose **Set up directory**.
2. Choose **Microsoft AD**.
3. Provide the following information:

   Edition

   Choose from either the **Standard** or **Enterprise** edition of AWS Microsoft AD. For more information about editions, see AWS Directory Service for Microsoft Active Directory (p. 1).

   Directory DNS

   The fully qualified name for the directory, such as `corp.example.com`.

NetBIOS name

The short name for the directory, such as `CORP`.

Administrator password

The password for the directory administrator. The directory creation process creates an administrator account with the user name `Admin` and this password.

The password cannot include the word "admin."

The directory administrator password is case-sensitive and must be between 8 and 64 characters in length, inclusive. It must also contain at least one character from three of the following four categories:

- Lowercase letters (a-z)
- Uppercase letters (A-Z)
- Numbers (0-9)
- Non-alphanumeric characters (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

Confirm password

Retype the administrator password.

Description

An optional description for the directory.

4. Provide the following information in the **VPC Details** section, and then choose **Next Step**.

   VPC

   The VPC for the directory.

   Subnets

   Select the subnets for the directory servers. The two subnets must be in different Availability Zones.
5. Review the directory information and make any necessary changes. When the information is correct, choose **Create Microsoft AD**.

It takes several minutes for the directory to be created. When it has been successfully created, the **Status** value changes to `Active`.

What Gets Created

When you create a directory with Microsoft AD, AWS Directory Service performs the following tasks on your behalf:

- Sets up Active Directory within the VPC running on two domain controllers for fault tolerance and high availability. If you need more domain controllers, you can add them later. For more information, see Deploy Additional Domain Controllers (p. 114).
- Creates a directory administrator account with the user name `Admin` and the specified password. You use this account to manage your directory

  Important
  Be sure to save this password. AWS Directory Service does not store this password and it cannot be retrieved or reset
- Creates a new **AWS Reserved** OU to store all AWS specific accounts
- Creates a security group for the domain controllers.

Admin Account

When you create an AWS Directory Service for Microsoft Active Directory directory, AWS creates an organizational unit (OU) that contains all your directory's objects. This OU, which has the NetBIOS name that you typed when you created your directory, is located in the domain root. The domain root is owned and managed by AWS.

The *admin* account that was created with your Microsoft AD has permissions for the most common administrative activities for your OU:

- Add, update, or delete users, groups, and computers. For more information, see Add Users and Groups (Simple AD and Microsoft AD) (p. 147).
- Add resources to your domain such as file or print servers, and then assign permissions for those resources to users and groups in your OU.
- Create additional OUs and containers.
- Delegate authority. For more information, see Overview of Managing Access Permissions to Your AWS Directory Service Resources (p. 176).
- Create and link group policies.
- Restore deleted objects from the Active Directory Recycle Bin.
- Run AD and DNS Windows PowerShell modules on the Active Directory Web Service.
- Create and configure group Managed Service Accounts. For more information, see Group Managed Service Accounts (p. 59).
- Configure Kerberos constrained delegation. For more information, see Kerberos Constrained Delegation (p. 60).

The admin account also has rights to perform the following domain-wide activities:

- Manage DNS configurations (Add, remove, or update records, zones and forwarders).
- View DNS event logs.
- View security event logs.

Actions not listed here are not allowed for the admin account. The admin account also lacks permissions for any directory-related actions outside of your specific OU, such as on the parent OU.

> **Important**
> AWS Domain Administrators have full administrative access to all domains hosted on AWS. See your agreement with AWS and the AWS Data Protection FAQ for more information about how AWS handles content, including directory information, that you store on AWS systems.

Group Managed Service Accounts

With Windows Server 2012, Microsoft introduced a new method that administrators could use to manage service accounts called group Managed Service Accounts (gMSAs). Using gMSAs, password synchronization between service instances no longer needed to be manually managed by service administrators. Instead, an admin could simply create a gMSA in AD and then configure multiple service instances to use that single gMSA.

To grant permissions so users in Microsoft AD can create a gMSA, you must add their accounts as a member of the *Managed Service Accounts Admins* security group. By default, the Admin account is a member of this group. For more information about gMSAs, see Group Managed Service Accounts Overview on the Microsoft TechNet website.

Kerberos Constrained Delegation

Kerberos constrained delegation is a feature in Windows Server that gives service administrators the ability to specify and enforce application trust boundaries by limiting the scope where application services can act on a user's behalf. This can be useful when you need to configure which front-end service accounts can delegate to their back-end services. Kerberos constrained delegation also prevents your gMSA from connecting to any and all services on behalf of your AD users, avoiding the potential for abuse by a rogue developer.

For example, let's say user jsmith logs into an HR application. You want the SQL Server to apply jsmith's database permissions. However, by default SQL Server opens the database connection using the service account credentials applying hr-app-service's permissions, instead of jsmith's configured permissions. To make it possible for the HR payroll application to access the SQL Server database using the jsmith's credentials you must enable Kerberos constrained delegation for the hr-app-service service account on your Microsoft AD directory in AWS. When jsmith logs on, Active Directory provides a Kerberos ticket that Windows automatically uses when jsmith attempts to access other services in the network. Kerberos delegation enables the hr-app-service account to reuse the jsmith Kerberos ticket when accessing the database, thus applying permissions specific to jsmith when opening the database connection.

To grant permissions so users in Microsoft AD can configure Kerberos constrained delegation, you must add their accounts as a member of the *Kerberos Delegations Admins* security group. By default, the Admin account is a member of this group. For more information about Kerberos constrained delegation, see Kerberos Constrained Delegation Overview on the Microsoft TechNet website.

Schema Extensions

A schema is the definition of attributes and classes that are part of a distributed directory and is similar to fields and tables in a database. Schemas include a set of rules which determine the type and format of data that can be added or included in the database. The User class is one example of a *class* that is stored in the database. Some example of User class attributes can include the user's first name, last name, phone number, and so on.

Microsoft AD uses schemas to organize and enforce how directory data is stored. The process of adding definitions to the schema is referred to as "extending the schema." Schema extensions make it possible for you to modify the schema of your Microsoft AD directory using a valid LDAP Data Interchange Format (LDIF) file. For more information about AD schemas and how to extend your schema, see the topics listed below.

Topics

Schema Elements

Attributes, classes and objects are the basic elements that are used to build object definitions in the schema. The following provides details about schema elements that are important to know before you begin the process to extend your Microsoft AD schema.

Attributes

Each schema attribute, which is similar to a field in a database, has several properties that define the characteristics of the attribute. For example, the property used by LDAP clients to read and write the attribute is `LDAPDisplayName`. The `LDAPDisplayName` property must be unique across all attributes and classes. For a complete list of attribute characteristics, see Characteristics of Attributes on the MSDN website. For additional guidance on how to create a new attribute, see Defining a New Attribute on the MSDN website.

Classes

The classes are analogous to tables in a database and also have several properties to be defined. For example, the `objectClassCategory` defines the class category. For a complete list of class characteristics, see Characteristics of Object Classes on the MSDN website. For more information about how to create a new class, see Defining a New Class on the MSDN website.

Object identifier (OID)

Each class and attribute must have an OID that is unique for all of your objects. Software vendors must obtain their own OID to ensure uniqueness. Uniqueness avoids conflicts when the same attribute is used by more than one application for different purposes. To ensure uniqueness, you can obtain a root OID from an ISO Name Registration Authority. Alternatively, you can obtain a base OID from Microsoft. For more information about OIDs and how to obtain them, see Object Identifiers on the MSDN website.

Schema linked attributes

Some attributes are linked between two classes with forward and back links. The best example is groups. When you look at a group it shows you the members of the group; if you look at a user you can see what groups it belongs to. When you add a user to a group, Active Directory creates a forward link to the group. Then Active Directory adds a back link from the group to the user. A unique link ID must be generated when creating an attribute that will be linked. For more information, see Linked Attributes on the MSDN website.

Related Topics

- When to Extend Your Microsoft AD Schema (p. 61)
- Tutorial: Extending Your Microsoft AD Schema (p. 62)

When to Extend Your Microsoft AD Schema

You can extend your Microsoft AD schema by adding new object classes and attributes. For example, you might do this if you have an application that requires changes to your schema in order to support single sign-on capabilities.

You can also use schema extensions to enable support for applications that rely on specific Active Directory object classes and attributes. This can be especially useful in the case where you need to migrate corporate applications that are dependent on Microsoft AD, to the AWS cloud.

Each attribute or class that is added to an existing Active Directory schema must be defined with a unique ID. That way when companies add extensions to the schema, they can be guaranteed to be unique and not to conflict with each other. These IDs are referred to as AD Object Identifiers (OIDs) and are stored in Microsoft AD.

To get started, see Tutorial: Extending Your Microsoft AD Schema (p. 62).

Related Topics

- Schema Extensions (p. 60)
- Schema Elements (p. 60)

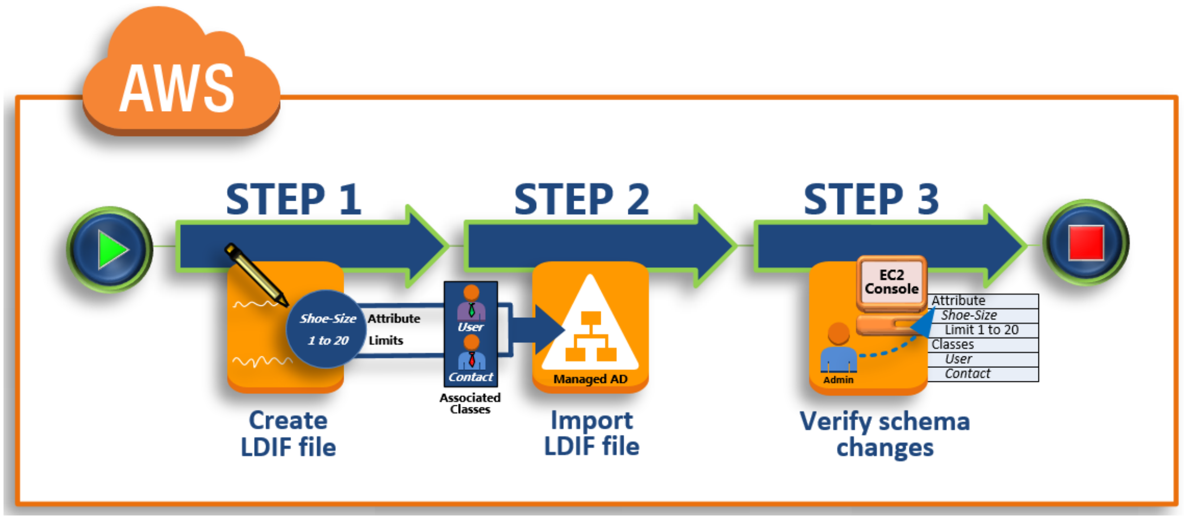
Tutorial: Extending Your Microsoft AD Schema

In this tutorial, you will learn how to extend the schema for your AWS Directory Service for Microsoft Active Directory directory, also known as Microsoft AD, by adding unique *attributes* and *classes* that meet your specific requirements. Microsoft AD schema extensions can only be uploaded and applied using a valid LDIF (Lightweight Directory Interchange Format) script file.

Attributes (attributeSchema) define the fields in the database while classes (classSchema) define the tables in the database. For example, all of the user objects in Active Directory are defined by the schema class *User* while the individual properties of a user, such as email address or phone number, are each defined by an attribute.

If you wanted to add a new property, such as Shoe-Size, you would define a new attribute, which would be of type *integer*. You could also define lower and upper limits like 1 to 20. Once the Shoe-Size attributeSchema object has been created, you would then alter the *User* classSchema object to contain that attribute. Attributes can be linked to multiple classes. Shoe-Size could also be added to the *Contact* class for example. For more information about Active Directory schemas, see When to Extend Your Microsoft AD Schema (p. 61).

This workflow has three basic steps.



Step 1: Create Your LDIF File (p. 62)

First, you create an LDIF file and define the new attributes and any classes that the attributes should be added to. You use this file for the next phase of the workflow.

Step 2: Import Your LDIF File (p. 64)

In this step, you use the AWS Directory Service console to import the LDIF file to your Microsoft AD environment.

Step 3: Verify If The Schema Extension Was Successful (p. 65)

Finally, as an administrator, you use an EC2 instance to verify that the new extensions appear in the Active Directory Schema Snap-in.

Step 1: Create Your LDIF File

An LDIF file is a standard plain text data interchange format for representing LDAP (Lightweight Directory Access Protocol) directory content and update requests. LDIF conveys directory content as a set of records, one record for each object (or entry). It also represents update requests, such as Add, Modify, Delete, and Rename, as a set of records, one record for each update request.

The AWS Directory Service imports your LDIF file with the schema changes by running the `ldifde.exe` application on your Microsoft AD directory. Therefore, you'll find it helpful to understand the LDIF script syntax. For more information, see LDIF Scripts.

Several third-party LDIF tools can extract, clean-up, and update your schema updates. Regardless of which tool you use, it is important to understand that all identifiers used in your LDIF file must be unique.

We highly recommend that you review the following concepts and tips prior to creating your LDIF file.

- **Schema elements** – Learn about schema elements such as attributes, classes, object IDs, and linked attributes. For more information, see Schema Elements (p. 60).

- **Sequence of items** – Make sure that the order in which the items in your LDIF file are laid out follow the Directory Information Tree (DIT) from the top down. The general rules for sequencing in an LDIF file include the following:
  - Separate items with a blank line.
  - List child items after their parent items.
  - Ensure that items such as attributes or object classes exist in the schema. If they are not present, you must add them to the schema before they can be used. For example, before you can assign an attribute to a class, the attribute must be created.
- **Format of the DN** – For each new instruction in the LDIF file, define the distinguished name (DN) as the first line of the instruction. The DN identifies an Active Directory object within the Active Directory object's tree and must contain the domain components for your directory. For example, the domain components for the directory in this tutorial are `DC=example,DC=com`.

  The DN also must contain the common name (CN) of the Active Directory object. The first CN entry is the attribute or class name. Next, you must use `CN=Schema,CN=Configuration`. This CN ensures that you are able to extend the Active Directory schema. As mentioned before, you cannot add or modify Active Directory objects' content. The general format for a DN follows.

```
dn: CN=[attribute or class name],CN=Schema,CN=Configuration,DC=[domain_name]
```

  For this tutorial, the DN for the new Shoe-Size attribute would look like:

```
dn: CN=Shoe-Size,CN=Schema,CN=Configuration,DC=example,DC=com
```

- **Warnings** – Review the warnings below before you extend your schema.
  - Before you extend your Active Directory schema, it is important to review Microsoft's warnings on the impact of this operation. For more information, see What You Must Know Before Extending the Schema.
  - You cannot delete a schema attribute or class. Therefore, if you make a mistake and don't want to restore from backup, you can only disable the object. For more information, see Disabling Existing Classes and Attributes.

To learn more about how LDIF files are constructed and see a sample LDIF file that can be used for testing Microsoft AD schema extensions, see the article How to Extend your Microsoft AD directory Schema on the AWS Security Blog.

Next Step

Step 2: Import Your LDIF File (p. 64)

Step 2: Import Your LDIF File

You can extend your schema by importing an LDIF file from either the AWS Directory Service console or by using the API. For more information about how to do this with the schema extension APIs, see the *AWS Directory Service API Reference*. At this time, AWS does not support external applications, such as Microsoft Exchange, to perform schema updates directly.

> **Important**
> When you make an update to your Microsoft AD directory schema, the operation is not reversible. In other words, once you create a new class or attribute, Active Directory doesn't allow you to remove it. However, you can disable it.
> If you must delete the schema changes, one option is to restore the directory from a previous snapshot. Restoring a snapshot rolls both the schema and the directory data back to a previous point, not just the schema.

Before the update process begins, Microsoft AD takes a snapshot to preserve the current state of your directory.

To import your LDIF file

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. In the **Directory ID** column, choose the link for your directory.
3. Under the **Schema extensions** tab, choose **Upload and update schema**.
4. In the dialog box, click **Browse**, select a valid LDIF file, type a description, and then choose **Update Schema**.

   > **Important**
   > Extending the schema is a critical operation. Don't apply any schema update in production environment without first testing it with your application in a development or test environment.

How is the LDIF File Applied

After your LDIF file has been uploaded, Microsoft AD takes steps to protect your directory against errors as it applies the changes in the following order.

1. **Validates the LDIF file.** Since LDIF scripts can manipulate any object in the domain, Microsoft AD runs checks right after you upload to help ensure that the import operation will not fail. These include checks to ensure the following:
   - The objects to be updated are only held in the schema container
   - The DC (domain controllers) part matches the name of the domain where the LDIF script is running
2. **Takes a snapshot of your directory.** You can use the snapshot to restore your directory in case you encounter any problems with your application after updating the schema.
3. **Applies the changes to a single DC.** Microsoft AD isolates one of your DCs and applies the updates in the LDIF file to the isolated DC. It then selects one of your DCs to be the schema master, removes that DC from directory replication, and applies your LDIF file using `Ldifde.exe`.
4. **Replication occurs to all DCs.** Microsoft AD adds the isolated DC back in to replication to complete the update. While this is all happening, your directory continues to provide the Active Directory service to your applications without disruption.

Next Step

Step 3: Verify If The Schema Extension Was Successful (p. 65)

Step 3: Verify If The Schema Extension Was Successful

After you have finished the import process, it is important to verify that schema updates were applied to your directory. This is especially critical before you migrate or update any application that relies on the schema update. You can do this using a variety of different LDAP tools or by writing a test tool that issues the appropriate LDAP commands.

This procedure uses the Active Directory Schema Snap-in and/or PowerShell to verify that the schema updates were applied. You must run these tools from a computer that is domain joined to your Microsoft AD. This can be a Windows server running in your on-premises network with access to your virtual private cloud (VPC) or through a virtual private network (VPN) connection. You can also run these tools on an Amazon EC2 Windows instance (see How to launch a new EC2 instance with Seamless Domain Join).

To verify using the Active Directory Schema Snap-in

1. Install the Active Directory Schema Snap-In using the instructions on the TechNet website.
2. Open the Microsoft Management Console (MMC) and expand the **AD Schema** tree for your directory.
3. Navigate through the **Classes** and **Attributes** folders until you find the schema changes that you made earlier.

To verify using PowerShell

1. Open a PowerShell window.
2. Use the `Get-ADObject` cmdlet as shown below to verify the schema change. For example:

```
get-adobject -Identity 'CN=Shoe-
Size,CN=Schema,CN=Configuration,DC=example,DC=com' -Properties *
```

Optional Step

(Optional) Add a value to the new attribute (p. 65)

(Optional) Add a value to the new attribute

Use this optional step when you have created a new attribute and want to add a new value to the attribute in your Microsoft AD directory.

To add a value to an attribute

1. Open the Windows PowerShell command line utility and set the new attribute with the following command. In this example, we will add a new EC2InstanceID value to the attribute for a specific computer.

```
PS C:\> set-adcomputer -Identity computer name -add @{example-EC2InstanceID
= 'EC2 instance ID'}
```

2. You can validate if the EC2InstanceID value was added to the computer object by running the following command:

```
PS C:\> get-adcomputer -Identity computer name -Property example-
EC2InstanceID
```

Related Resources

The following resource links are located on the Microsoft website and provide related information.

- Extending the Schema (Windows)
- Active Directory Schema (Windows)
- Active Directory Schema
- Windows Administration: Extending the Active Directory Schema
- Restrictions on Schema Extension (Windows)
- Ldifde

When to Create a Trust Relationship

You can configure one and two-way forest trust relationships between your AWS Directory Service for Microsoft Active Directory and on-premises directories, as well as between multiple Microsoft AD directories in the AWS cloud. Microsoft AD supports all three trust relationship directions: Incoming, Outgoing and Two-way (Bi-directional).

> **Note**
> When setting up trust relationships, you must ensure that your on-premises directory is and remains compatible with AWS Directory Services. For more information on your responsibilities, please see our shared responsibility model.

Microsoft AD supports forest trusts only. External domain trusts and name suffix routing are not supported. To walk through an example scenario showing how to create a trust, see Tutorial: Create a Trust Relationship Between Your Microsoft AD and Your On-Premises Domain (p. 73).

Prerequisites

Creating the trust requires only a few steps, but you must first complete several prerequisite steps prior to setting up the trust.

Connect to VPC

If you are creating a trust relationship with your on-premises directory, you must first connect your on-premises network to the VPC containing your Microsoft AD. The firewall for your on-premises network must have the following ports open to the CIDRs for both subnets in the VPC.

- TCP/UDP 53 - DNS
- TCP/UDP 88 - Kerberos authentication
- TCP/UDP 389 - LDAP
- TCP 445 - SMB

These are the minimum ports that are needed to be able to connect to your directory. Your specific configuration may require additional ports be open.

Configure your VPC

The VPC that contains your Microsoft AD must have the appropriate outbound and inbound rules.

To configure your VPC outbound rules

1. In the AWS Directory Service console, on the **Directory Details** page, note your Microsoft AD directory ID.
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.

3. Choose **Security Groups**.
4. Search for your Microsoft AD directory ID. In the search results, select the item with the description "AWS created security group for *directory ID* directory controllers".

   Note
   The selected security group is a security group that is automatically created when you initially create your directory.
5. Go to the **Outbound Rules** tab of that security group. Select **Edit**, then **Add another rule**. For the new rule, enter the following values:

   - **Type**: All Traffic
   - **Protocol**: All
   - **Destination** determines the traffic that can leave your domain controllers and where it can go in your on-premises network. Specify a single IP address or an IP address range in CIDR notation (for example, 203.0.113.5/32). You can also specify the name or ID of another security group in the same region. For more information, see Understand Your Directory's AWS Security Group Configuration and Use (p. 8).
6. Select **Save**.

To configure your VPC inbound rules

1. In the AWS Directory Service console, on the **Directory Details** page, note your Microsoft AD directory ID.
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. Choose **Security Groups**.
4. Search for your Microsoft AD directory ID. In the search results, select the item with the description "AWS created security group for *directory ID* directory controllers".

   Note
   The selected security group is a security group that is automatically created when you initially create your directory.
5. Go to the **Inbound Rules** tab of that security group. Select **Edit**, then **Add another rule**. For the new rule, enter the following values:

   - **Type**: Custom UDP Rule
   - **Protocol**: UDP
   - **Port Range**: 445
   - For **Source**, specify a single IP address, or an IP address range in CIDR notation (for example, 203.0.113.5/32). You can also specify the name or ID of another security group in the same region. This setting determines the traffic that can reach your domain controllers from your on-premises network. For more information, see Understand Your Directory's AWS Security Group Configuration and Use (p. 8).
6. Select **Save**.
7. Repeat these steps, adding each of the following rules:

Type	Protocol	Port Range	Source
Custom UDP Rule	UDP	88	Specify the **Source** traffic used in the previous step.

Type	Protocol	Port Range	Source
Custom UDP Rule	UDP	123	Specify the **Source** traffic used in the previous step.
Custom UDP Rule	UDP	138	Specify the **Source** traffic used in the previous step.
Custom UDP Rule	UDP	389	Specify the **Source** traffic used in the previous step.
Custom UDP Rule	UDP	464	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	88	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	135	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	445	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	464	Specify the **Source** traffic used in the previous step.

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	636	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	1024 - 65535	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	3268 - 3269	Specify the **Source** traffic used in the previous step.
DNS (UDP)	UDP	53	Specify the **Source** traffic used in the previous step.
DNS (TCP)	TCP	53	Specify the **Source** traffic used in the previous step.
LDAP	TCP	389	Specify the **Source** traffic used in the previous step.
All ICMP	All	N/A	Specify the **Source** traffic used in the previous step.
All traffic	All	All	The current security group (The security group for your directory).

These security rules impact an internal network interface that is not exposed publicly.

Enable Kerberos Pre-authentication

Your user accounts must have Kerberos pre-authentication enabled. For more information about this setting, review Preauthentication on Microsoft TechNet.

Configure DNS Conditional Forwarders On Your On-premises domain

You must set up DNS conditional forwarders on your on-premises domain. Refer to Assign a Conditional Forwarder for a Domain Name on Microsoft TechNet for details on conditional forwarders.

To perform the following steps, you must have access to following Windows Server tools for your on-premises domain:

- AD DS and AD LDS Tools
- DNS

To configure conditional forwarders on your on-premises domain

1. First you must get some information about your AWS Microsoft AD. Sign into the AWS Management Console and open the AWS Directory Service console at https://console.aws.amazon.com/directoryservice/.
2. In the navigation pane, select **Directories**.
3. Choose the directory ID of your Microsoft AD.
4. Take note of the fully qualified domain name (FQDN) and the DNS addresses of your directory.
5. Now, return to your on-premises domain controller. Open Server Manager.
6. On the **Tools** menu, choose **DNS**.
7. In the console tree, expand the DNS server of the domain for which you are setting up the trust.
8. In the console tree, choose **Conditional Forwarders**.
9. On the **Action** menu, choose **New conditional forwarder**.
10. In **DNS domain**, type the fully qualified domain name (FQDN) of your Microsoft AD, which you noted earlier.
11. Choose **IP addresses of the master servers** and type the DNS addresses of your Microsoft AD directory, which you noted earlier.

    After entering the DNS addresses, you might get a "timeout" or "unable to resolve" error. You can generally ignore these errors.
12. Select **Store this conditional forwarder in Active Directory and replicate as follows: All DNS servers in this domain**. Choose **OK**.

Trust Relationship Password

If you are creating a trust relationship with an existing domain, set up the trust relationship on that domain using Windows Server Administration tools. As you do so, note the trust password that you use. You will need to use this same password when setting up the trust relationship on the Microsoft AD. For more information, see Managing Trusts on Microsoft TechNet.

You are now ready to create the trust relationship on your Microsoft AD.

Create, Verify, or Delete a Trust Relationship

To create a trust relationship with your Microsoft AD

1. Open the AWS Directory Service console.
2. Choose the directory you want to configure.
3. On the **Details** page, choose the **Trusts** tab.
4. Choose **Add Trust Relationship**.
5. Provide the required information, including the fully qualified domain name (FQDN) of your trusted domain, the trust password and the trust direction.
6. For **Conditional forwarder**, type the IP address of your on-premises DNS server. If you have previously created conditional forwarders, you can type the fully qualified domain name (FQDN) of your on-premises domain instead of a DNS IP address.
7. (Optional) Choose **Add IP address** and type the IP address of an additional on-premises DNS server. You can repeat this step for each applicable DNS server address for a total of four addresses.
8. Choose **Create**.
9. If the DNS server for your on-premises domain uses a publicly addressable IP address, choose the **IP routing** tab and choose **Add route**. Type the IP address block of your DNS server using CIDR format, for example 10.0.0.0/24. This step is not necessary if your DNS server does not use a public IP address.
10. (Optional) We recommend that you also select **Add routes to the security group for this directory's VPC**. This will configure the security groups as detailed above in the "Configure your VPC." These security rules impact an internal network interface that is not exposed publicly. If this option is not available, you will instead see a message indicating that you have already customized your security groups.

You must set up the trust relationship on both domains. The relationships must be complementary. For example, if you create an outgoing trust on one domain, you must create an incoming trust on the other.

If you are creating a trust relationship with an existing domain, set up the trust relationship on that domain using Windows Server Administration tools.

You can create multiple trusts between your Microsoft AD and various Active Directory domains. However, only one trust relationship per pair can exist at a time. For example, if you have an existing, one-way trust in the “Incoming direction” and you then want to set up another trust relationship in the “Outgoing direction,” you will need to delete the existing trust relationship, and create a new “Two-way” trust.

To verify an outgoing trust relationship

1. Open the AWS Directory Service console.
2. Choose the directory you wish to configure.
3. On the **Details** page, choose the **Trusts** tab.
4. Choose the trust relationship to verify.
5. For **Actions**, choose **Verify**.

This process verifies only the outgoing direction of a two-way trust. AWS does not support verification of an incoming trusts. For more information on how to verify a trust to or from your on-premises Active Directory, refer to Verify a Trust on Microsoft Technet.

To delete an existing trust relationship

1. Open the AWS Directory Service console.

2. Choose the directory you wish to configure.
3. On the **Details** page, choose the **Trusts** tab.
4. Choose the trust relationship to delete.
5. For **Actions**, choose **Delete**.

Adding IP Routes When Using Public IP Addresses

You can use AWS Directory Service for Microsoft Active Directory to take advantage of many powerful Active Directory features, including establishing trusts with other directories. However, if the DNS servers for the other directories use public IP addresses, you must specify those IP addresses as part of configuring the trust. Instructions for doing this can be found in When to Create a Trust Relationship (p. 66).

Similarly, you must also enter the IP address information when routing traffic from your Microsoft AD on AWS to a peer AWS VPC, if the VPC uses public IP ranges.

When you add the IP addresses as described in When to Create a Trust Relationship (p. 66), you have the option of selecting **Add routes to the security group for this directory's VPC**. This option should be selected unless you have previously customized your security group to allow the necessary traffic as shown below. For more information, see Understand Your Directory's AWS Security Group Configuration and Use (p. 8).

This option configures the security groups for your directory's VPC as follows:

Inbound rules

Type	Protocol	Port Range	Source
Custom UDP Rule	UDP	88	*0.0.0.0/0*
Custom UDP Rule	UDP	123	*0.0.0.0/0*
Custom UDP Rule	UDP	138	*0.0.0.0/0*
Custom UDP Rule	UDP	389	*0.0.0.0/0*
Custom UDP Rule	UDP	445	*0.0.0.0/0*
Custom UDP Rule	UDP	464	*0.0.0.0/0*
Custom TCP Rule	TCP	88	*0.0.0.0/0*
Custom TCP Rule	TCP	135	*0.0.0.0/0*
Custom TCP Rule	TCP	445	*0.0.0.0/0*
Custom TCP Rule	TCP	464	*0.0.0.0/0*
Custom TCP Rule	TCP	636	*0.0.0.0/0*
Custom TCP Rule	TCP	1024 - 65535	*0.0.0.0/0*
Custom TCP Rule	TCP	3268 - 3269	*0.0.0.0/0*
DNS (UDP)	UDP	53	*0.0.0.0/0*
DNS (TCP)	TCP	53	*0.0.0.0/0*
LDAP	TCP	389	*0.0.0.0/0*

Type	Protocol	Port Range	Source
All ICMP	All	N/A	*0.0.0.0/0*

Outbound rules

Type	Protocol	Port Range	Destination
All traffic	All	All	*0.0.0.0/0*

These security rules affect an internal network interface that is not exposed publicly.

Tutorial: Create a Trust Relationship Between Your Microsoft AD and Your On-Premises Domain

This tutorial walks you through all the steps necessary to set up a trust relationship between AWS Directory Service for Microsoft Active Directory and your on-premises Microsoft Active Directory. Although creating the trust requires only a few steps, you must first complete the following prerequisite steps.

Topics

See Also

When to Create a Trust Relationship (p. 66)

Prerequisites

This tutorial assumes you already have the following:

- A Microsoft AD created on AWS. If you need help doing this, see Create a Microsoft AD Directory (p. 56).
- An EC2 instance running Windows added to that Microsoft AD. If you need help doing this, see Manually Add a Windows Instance (Simple AD and Microsoft AD) (p. 157).

  Important
  The admin account for your Microsoft AD must have administrative access to this instance.
- The following Windows Server tools installed on that instance:
  - AD DS and AD LDS Tools
  - DNS

  If you need help doing this, see Installing the Active Directory Administration Tools (p. 148).
- An on-premises Microsoft Active Directory

  You must have administrative access to this directory. The same Windows Server tools as listed above must also be available for this directory.

- An active connection between your on-premises network and the VPC containing your Microsoft AD. If you need help doing this, see Amazon Virtual Private Cloud Connectivity Options.

Tutorial Configuration

For this tutorial, we've already created a Microsoft AD and an on-premises domain. The on-premises network is connected to the Microsoft AD's VPC. Following are the properties of the two directories:

Microsoft AD running on AWS

- Domain name (FQDN): MyManagedAD.example.com
- NetBIOS name: MyManagedAD
- DNS Addresses: 10.0.10.246, 10.0.20.121
- VPC CIDR: 10.0.0.0/16

The Microsoft AD resides in VPC ID: vpc-12345678.

On-premises domain

- Domain name (FQDN): corp.example.com
- NetBIOS name: CORP
- DNS Addresses: 172.16.10.153
- On-premises CIDR: 172.16.0.0/16

Next Step

Step 1: Prepare Your On-Premises Domain (p. 74)

Step 1: Prepare Your On-Premises Domain

First you need to complete several prerequisite steps on your on-premises domain.

Configure Your On-Premises Firewall

You must configure your on-premises firewall so that the following ports are open to the CIDRs for all subnets used by the VPC that contains your Microsoft AD. In this tutorial, we allow both incoming and outgoing traffic from 10.0.0.0/16 (the CIDR block of our Microsoft AD's VPC) on the following ports:

- TCP/UDP 53 - DNS
- TCP/UDP 88 - Kerberos authentication
- TCP/UDP 389 - LDAP
- TCP 445 - SMB

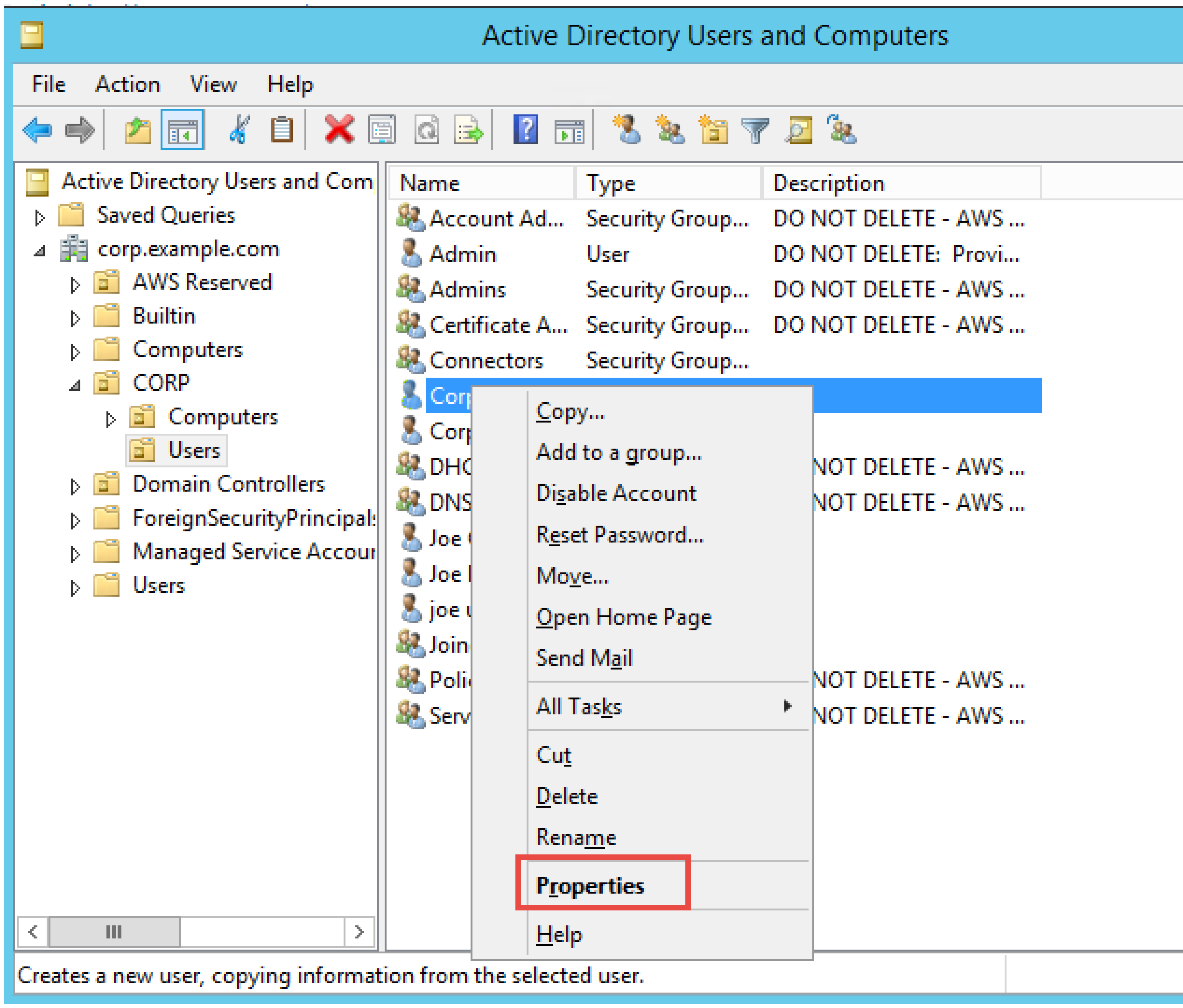
Note
These are the minimum ports that are needed to connect the VPC to the on-premises directory. Your specific configuration may require additional ports be open.

Ensure That Kerberos Pre-authentication Is Enabled

User accounts in both directories must have Kerberos preauthentication enabled. This is the default, but let's check to make sure nothing has changed.

To view user Kerberos settings

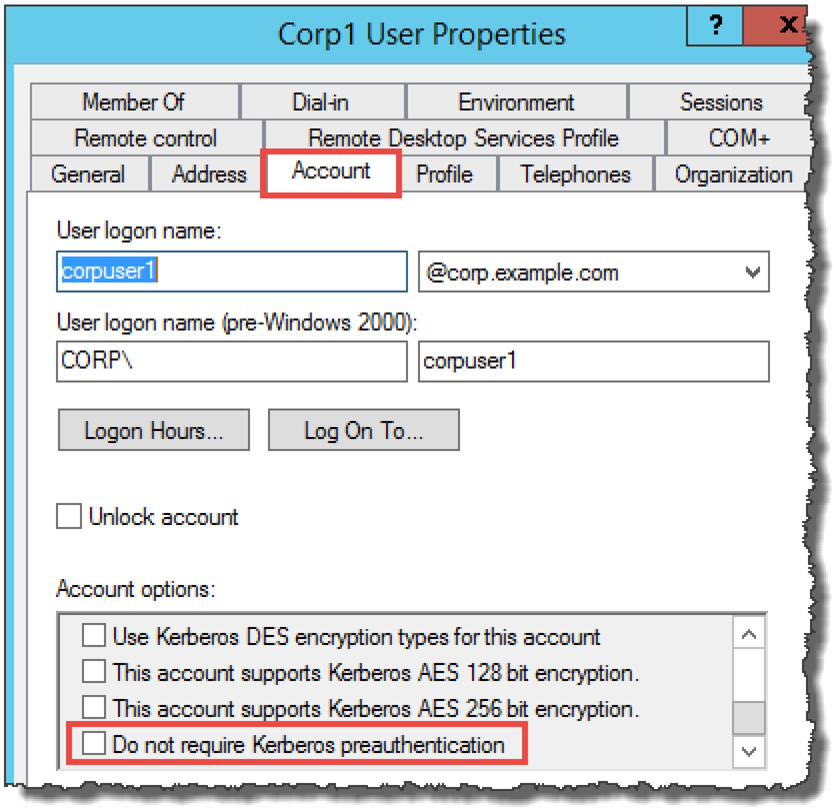
1. On your on-premises domain controller, open Server Manager.
2. On the **Tools** menu, choose **Active Directory Users and Computers**.
3. Choose the **Users** folder and open the context (right-click) menu for a user account listed in the right pane. Choose **Properties**.



Name	Type	Description
Account Ad...	Security Group...	DO NOT DELETE - AWS ...
Admin	User	DO NOT DELETE: Provi...
Admins	Security Group...	DO NOT DELETE - AWS ...
Certificate A...	Security Group...	DO NOT DELETE - AWS ...
Connectors	Security Group...	
Corp		
Corp		
DHC		NOT DELETE - AWS ...
DNS		NOT DELETE - AWS ...
Joe		
Joe		
joe		
Join		
Poli		NOT DELETE - AWS ...
Serv		NOT DELETE - AWS ...

Creates a new user, copying information from the selected user.

4. Choose the **Account** tab. In the **Account options** list, scroll down and ensure that **Do not require Kerberos preauthentication** is *not* checked.



User logon name:
corpuser1 @corp.example.com

User logon name (pre-Windows 2000):
CORP\ corpuser1

☐ Unlock account

Account options:

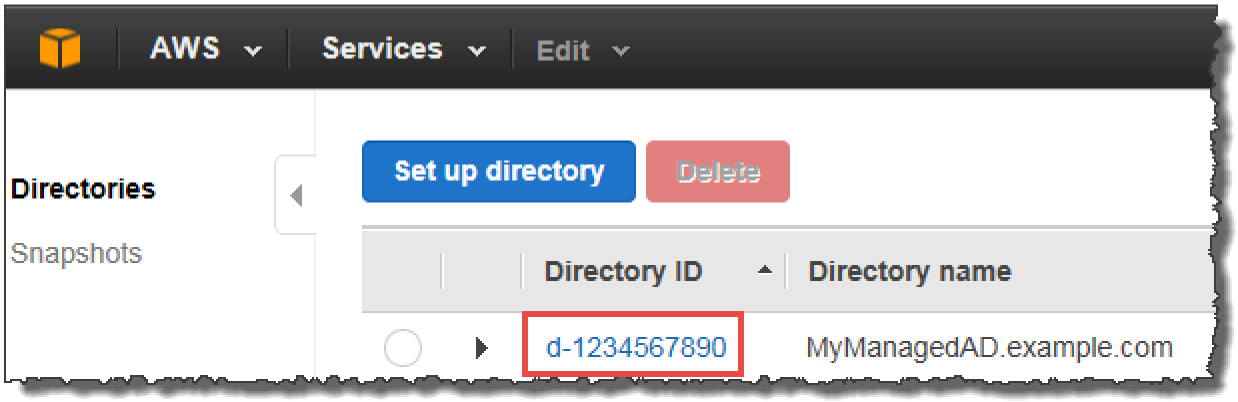
☐ Use Kerberos DES encryption types for this account
☐ This account supports Kerberos AES 128 bit encryption.
☐ This account supports Kerberos AES 256 bit encryption.
☐ Do not require Kerberos preauthentication

Configure DNS Conditional Forwarders for Your On-premises Domain

You must set up DNS conditional forwarders on each domain. Before doing this on your on-premises domain, you will first get some information about your AWS Microsoft AD.

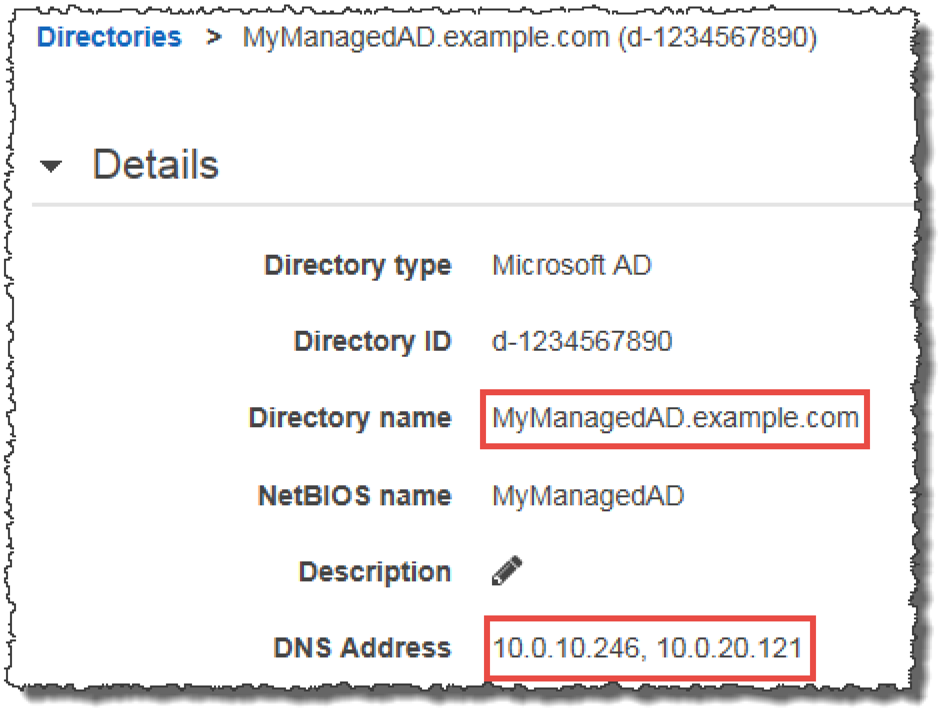
To configure conditional forwarders on your on-premises domain

1. Sign into the AWS Management Console and open the AWS Directory Service console at https://console.aws.amazon.com/directoryservice/.
2. In the navigation pane, select **Directories**.
3. Choose the directory ID of your Microsoft AD.



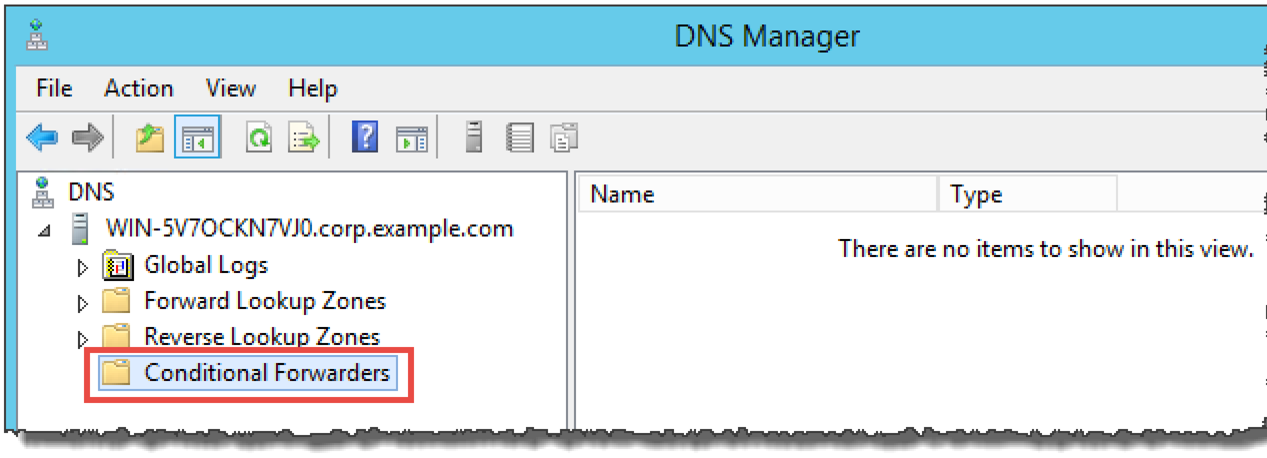
Directory ID	Directory name
d-1234567890	MyManagedAD.example.com

4. Take note of the fully qualified domain name (FQDN) and the DNS addresses of your directory.

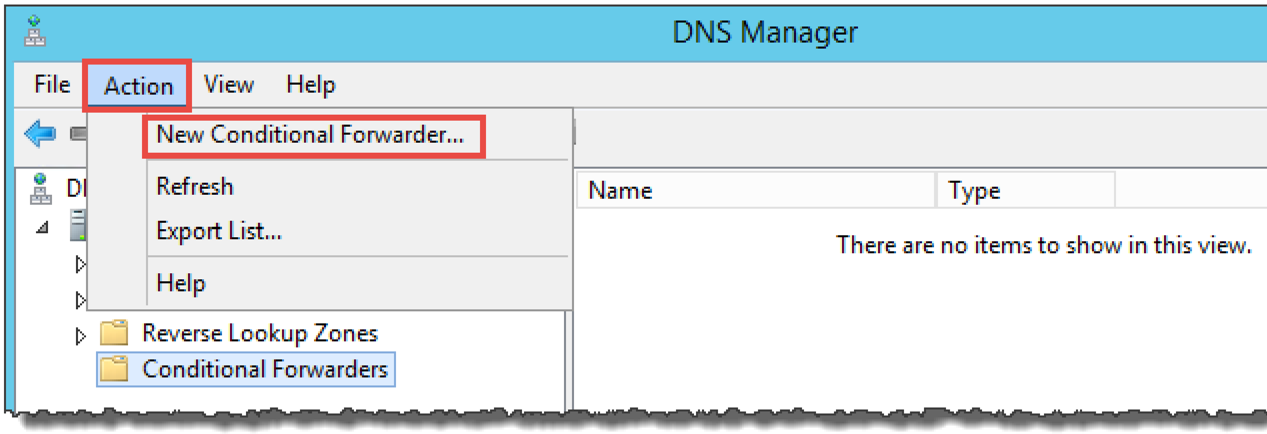


Directory type	Microsoft AD
Directory ID	d-1234567890
Directory name	MyManagedAD.example.com
NetBIOS name	MyManagedAD
Description	
DNS Address	10.0.10.246, 10.0.20.121

5. Now, return to your on-premises domain controller. Open Server Manager.
6. On the **Tools** menu, choose **DNS**.
7. In the console tree, expand the DNS server of the domain for which you are setting up the trust. Our server is WIN-5V70CN7VJ0.corp.example.com.
8. In the console tree, choose **Conditional Forwarders**.



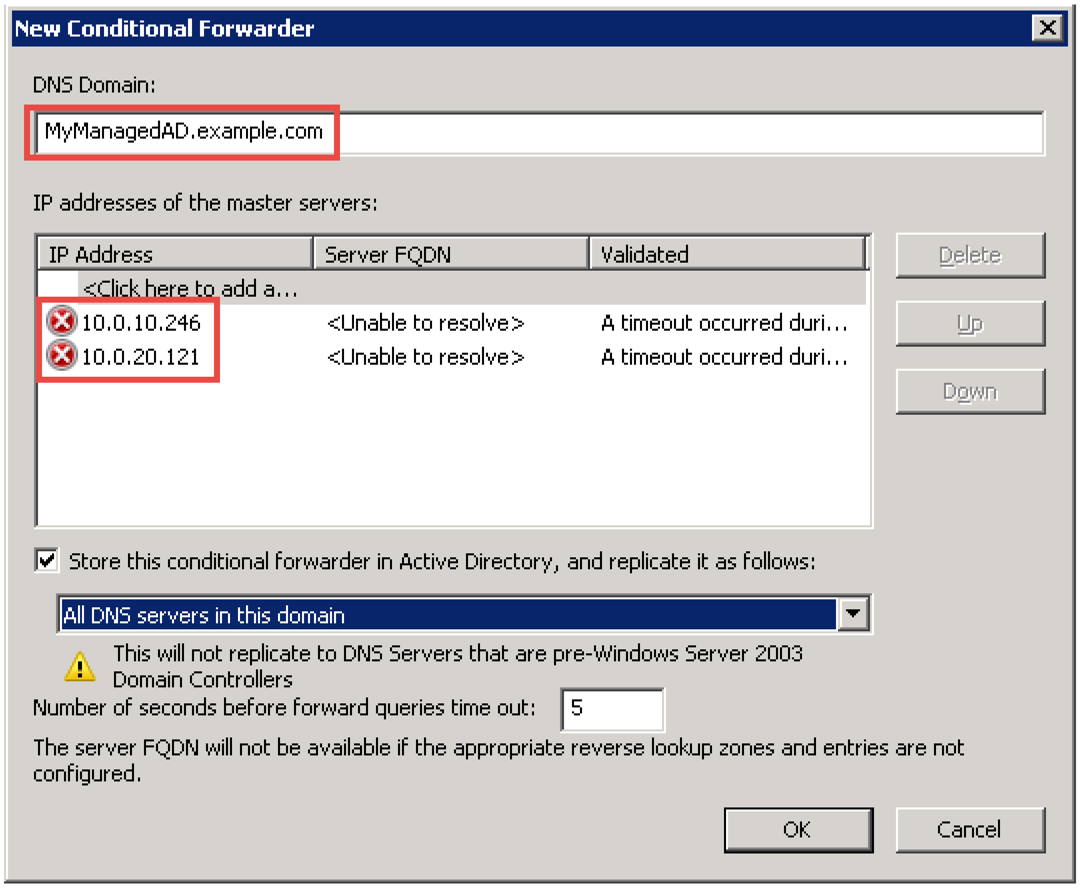
9. On the **Action** menu, choose **New conditional forwarder**.



10. In **DNS domain**, type the fully qualified domain name (FQDN) of your Microsoft AD, which you noted earlier. In this example, the FQDN is MyManagedAD.example.com.

11. Choose **IP addresses of the master servers** and type the DNS addresses of your Microsoft AD directory, which you noted earlier. In this example those are: 10.0.10.246, 10.0.20.121

    After entering the DNS addresses, you might get a "timeout" or "unable to resolve" error. You can generally ignore these errors.


New Conditional Forwarder

DNS Domain: MyManagedAD.example.com

IP addresses of the master servers:

IP Address	Server FQDN	Validated
<Click here to add a...		
10.0.10.246	<Unable to resolve>	A timeout occurred duri...
10.0.20.121	<Unable to resolve>	A timeout occurred duri...

☑ Store this conditional forwarder in Active Directory, and replicate it as follows:

All DNS servers in this domain

This will not replicate to DNS Servers that are pre-Windows Server 2003 Domain Controllers

Number of seconds before forward queries time out: 5

The server FQDN will not be available if the appropriate reverse lookup zones and entries are not configured.

12. Select **Store this conditional forwarder in Active Directory, and replicate it as follows**.
13. Select **All DNS servers in this domain**, and then choose **OK**.

Next Step

Step 2: Prepare Your Microsoft AD (p. 79)

Step 2: Prepare Your Microsoft AD

Now let's get your Microsoft AD ready for the trust relationship. Many of the following steps are almost identical to what you just completed for your on-premises domain. This time, however, you are working with your Microsoft AD.

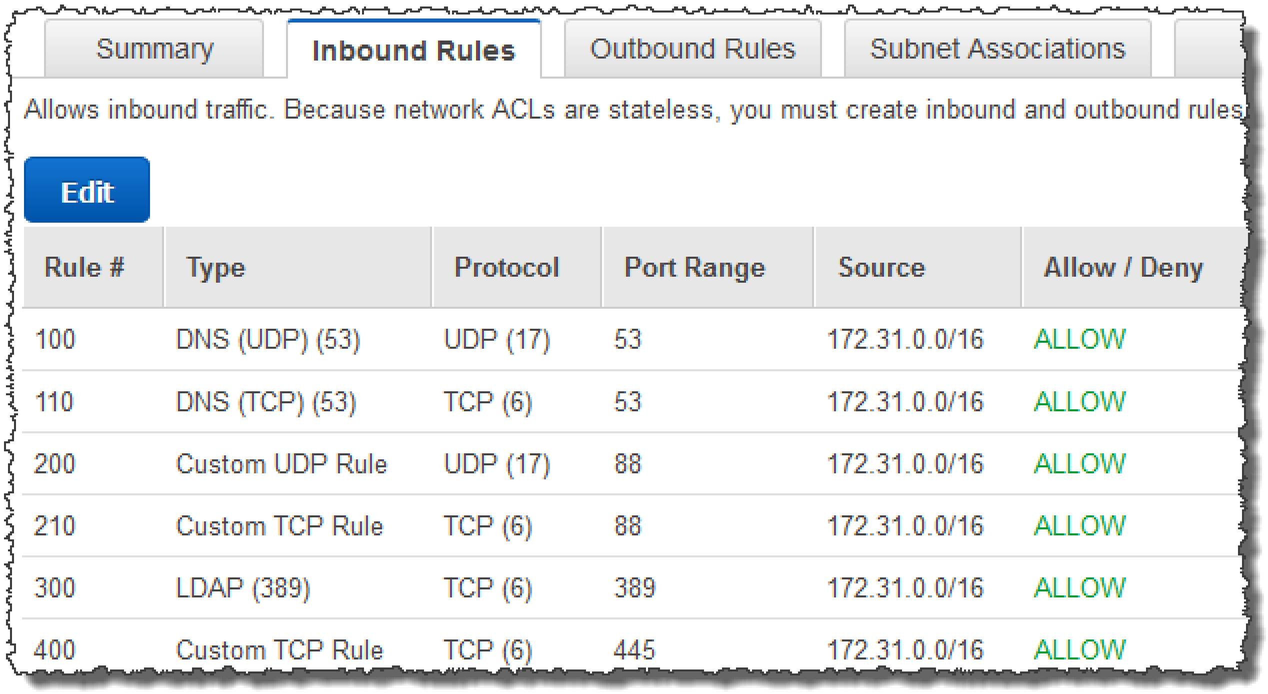
Configure Your VPN Subnets and Security Groups

You must allow traffic from your on-premises network to the VPC containing your Microsoft AD. To do this, configure the VPC access control list (ACL) to allow both incoming and outgoing traffic from your on-premises directory for the following ports:

- TCP/UDP 53 - DNS
- TCP/UDP 88 - Kerberos authentication
- TCP/UDP 389 - LDAP
- TCP 445 - SMB

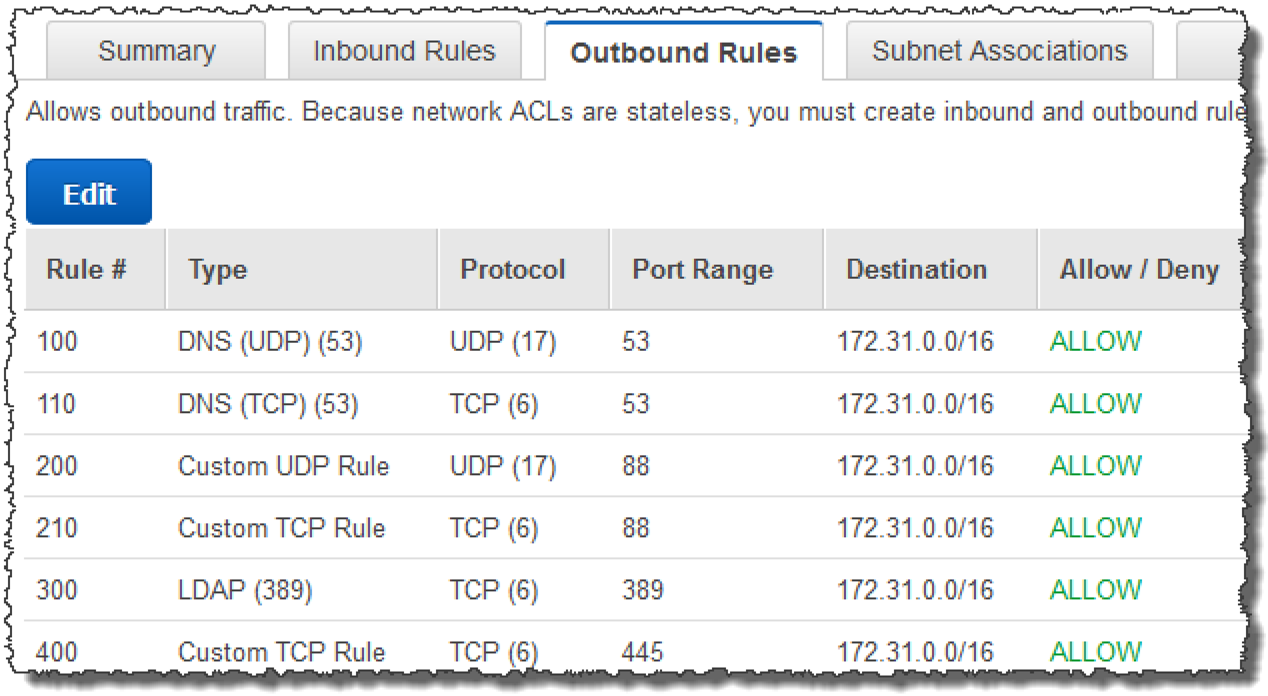
Note

These are the minimum ports that are needed to be able to connect the VPC and on-premises directory. Your specific configuration may require additional ports be open. For this tutorial, we have opened up all ports to our on-premises domain:



Allows inbound traffic. Because network ACLs are stateless, you must create inbound and outbound rules

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	DNS (UDP) (53)	UDP (17)	53	172.31.0.0/16	ALLOW
110	DNS (TCP) (53)	TCP (6)	53	172.31.0.0/16	ALLOW
200	Custom UDP Rule	UDP (17)	88	172.31.0.0/16	ALLOW
210	Custom TCP Rule	TCP (6)	88	172.31.0.0/16	ALLOW
300	LDAP (389)	TCP (6)	389	172.31.0.0/16	ALLOW
400	Custom TCP Rule	TCP (6)	445	172.31.0.0/16	ALLOW



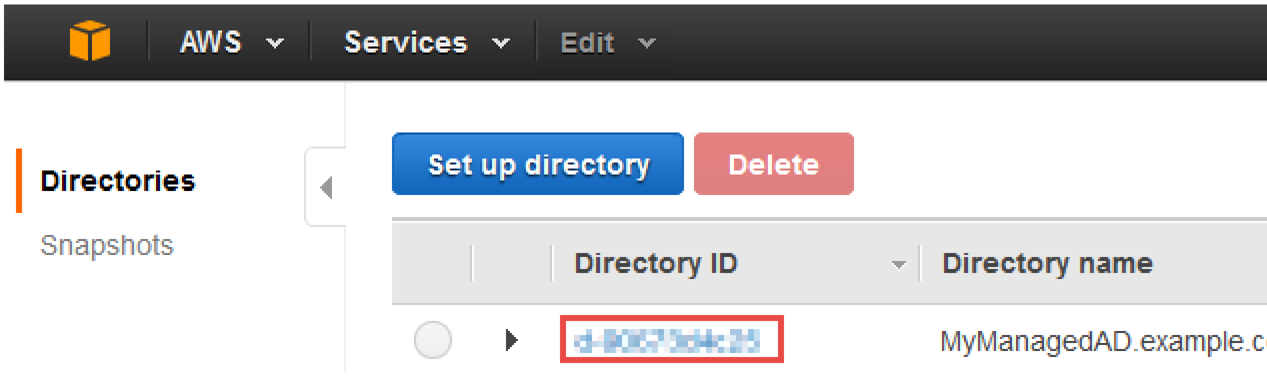
Allows outbound traffic. Because network ACLs are stateless, you must create inbound and outbound rule

Rule #	Type	Protocol	Port Range	Destination	Allow / Deny
100	DNS (UDP) (53)	UDP (17)	53	172.31.0.0/16	ALLOW
110	DNS (TCP) (53)	TCP (6)	53	172.31.0.0/16	ALLOW
200	Custom UDP Rule	UDP (17)	88	172.31.0.0/16	ALLOW
210	Custom TCP Rule	TCP (6)	88	172.31.0.0/16	ALLOW
300	LDAP (389)	TCP (6)	389	172.31.0.0/16	ALLOW
400	Custom TCP Rule	TCP (6)	445	172.31.0.0/16	ALLOW

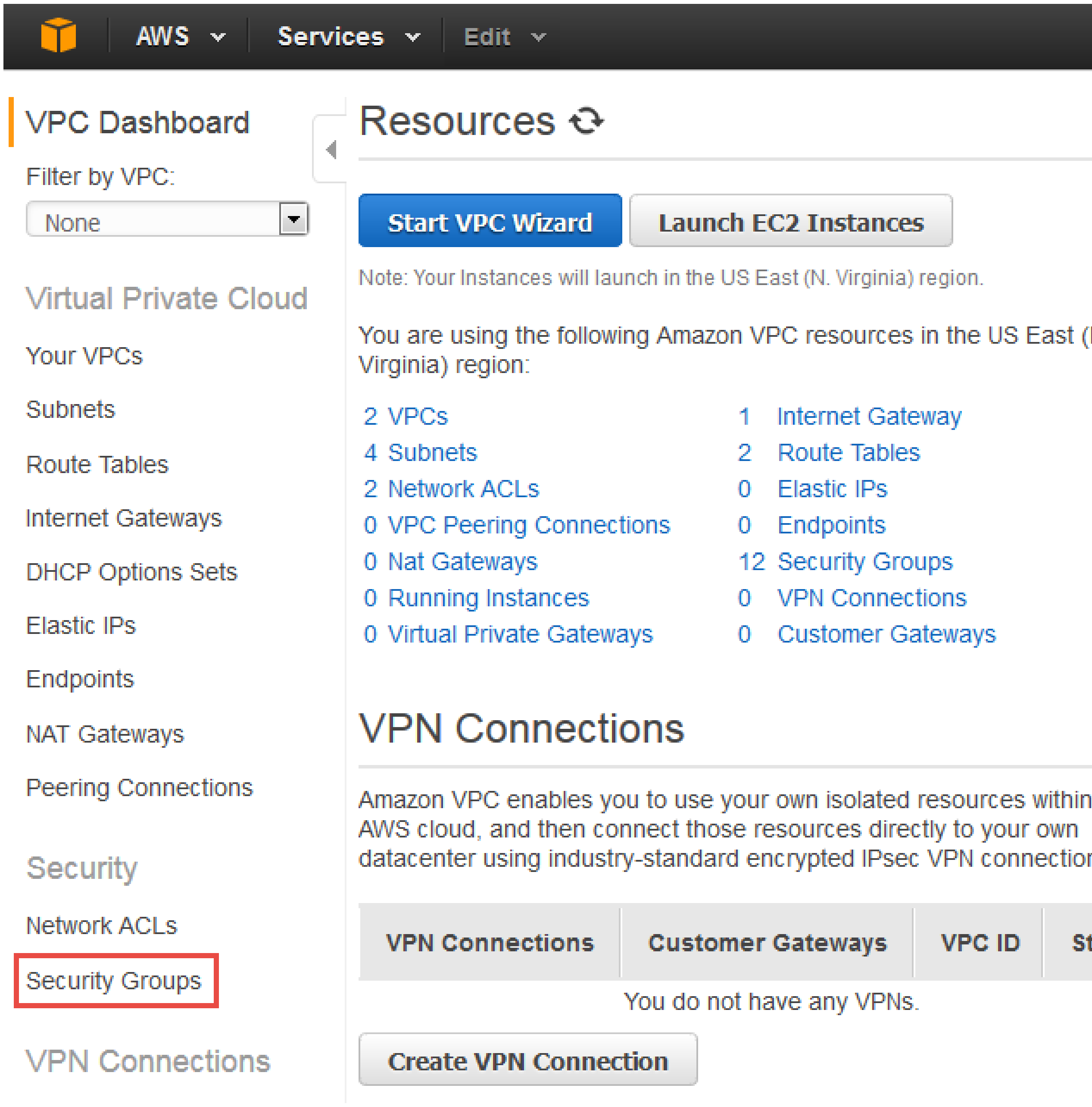
Similarly, your Microsoft AD domain controller must have the appropriate outbound and inbound rules.

To configure your Microsoft AD domain controller outbound and inbound rules

1. Return to the AWS Directory Service console at https://console.aws.amazon.com/directoryservice/. On the **Directory Details** page, note your Microsoft AD directory ID.



2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. In the navigation pane, choose **Security Groups**.



Note: Your Instances will launch in the US East (N. Virginia) region.

You are using the following Amazon VPC resources in the US East (
Virginia) region:

2 VPCs	1 Internet Gateway
4 Subnets	2 Route Tables
2 Network ACLs	0 Elastic IPs
0 VPC Peering Connections	0 Endpoints
0 Nat Gateways	12 Security Groups
0 Running Instances	0 VPN Connections
0 Virtual Private Gateways	0 Customer Gateways

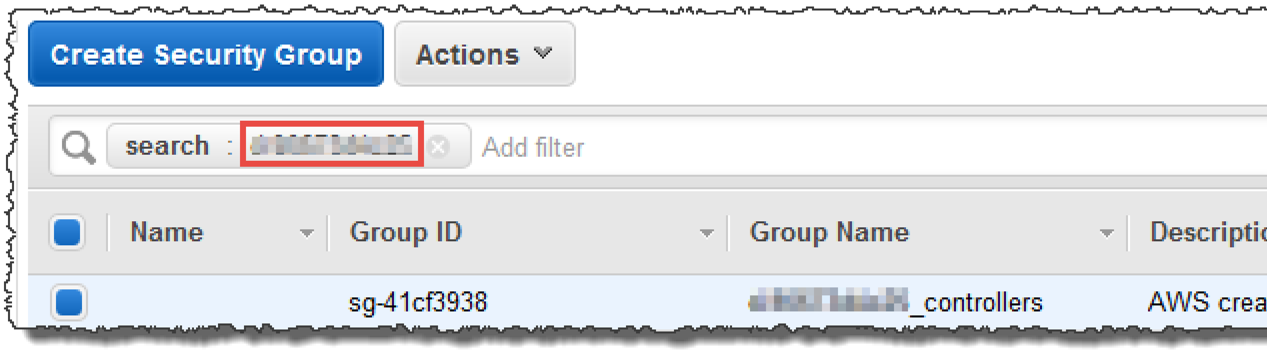
VPN Connections

Amazon VPC enables you to use your own isolated resources withi
AWS cloud, and then connect those resources directly to your own
datacenter using industry-standard encrypted IPsec VPN connectio

VPN Connections	Customer Gateways	VPC ID	St

You do not have any VPNs.

4. Use the search box to search for your Microsoft AD directory ID. In the search results, select the item with the description **AWS created security group for <yourdirectoryID> directory controllers**.

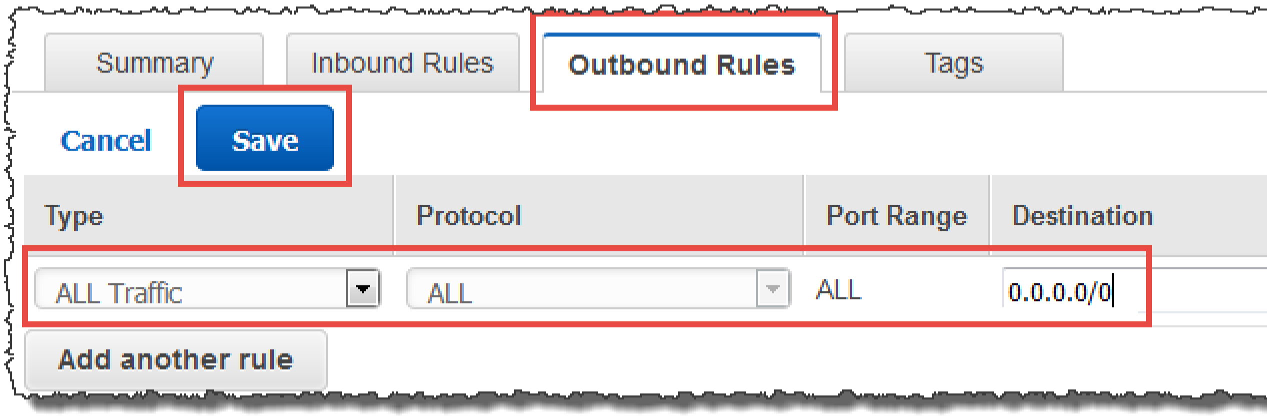


Name	Group ID	Group Name	Descripti
	sg-41cf3938	[redacted]_controllers	AWS crea

5. Go to the **Outbound Rules** tab for that security group. Choose **Edit**, and then **Add another rule**. For the new rule, enter the following values:

   - **Type**: ALL Traffic
   - **Protocol**: ALL
   - **Destination** determines the traffic that can leave your domain controllers and where it can go. Specify a single IP address or an IP address range in CIDR notation (for example, 203.0.113.5/32). You can also specify the name or ID of another security group in the same region. For more information, see Understand Your Directory's AWS Security Group Configuration and Use (p. 8).

6. Select **Save**.



Type	Protocol	Port Range	Destination
ALL Traffic	ALL	ALL	0.0.0.0/0

7. Go to the **Inbound Rules** tab for that same security group. Choose **Edit**, and then **Add another rule**. For the new rule, enter the following values:

   - **Type**: Custom UDP Rule
   - **Protocol**: UDP
   - **Port Range**: 445
   - For **Source**, specify a single IP address, or an IP address range in CIDR notation (for example, 203.0.113.5/32). You can also specify the name or ID of another security group in the same region. This setting determines the traffic that can reach your domain controllers. For more information, see Understand Your Directory's AWS Security Group Configuration and Use (p. 8).

8. Select **Save**.
9. Repeat these steps, adding each of the following rules:

Type	Protocol	Port Range	Source
Custom UDP Rule	UDP	88	Specify the **Source** traffic used in the previous step.
Custom UDP Rule	UDP	123	Specify the **Source** traffic used in the previous step.
Custom UDP Rule	UDP	138	Specify the **Source** traffic used in the previous step.
Custom UDP Rule	UDP	389	Specify the **Source** traffic used in the previous step.
Custom UDP Rule	UDP	464	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	88	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	135	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	445	Specify the **Source** traffic used in the previous step.

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	464	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	636	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	1024 - 65535	Specify the **Source** traffic used in the previous step.
Custom TCP Rule	TCP	3268 - 3269	Specify the **Source** traffic used in the previous step.
DNS (UDP)	UDP	53	Specify the **Source** traffic used in the previous step.
DNS (TCP)	TCP	53	Specify the **Source** traffic used in the previous step.
LDAP	TCP	389	Specify the **Source** traffic used in the previous step.
All ICMP	All	N/A	Specify the **Source** traffic used in the previous step.

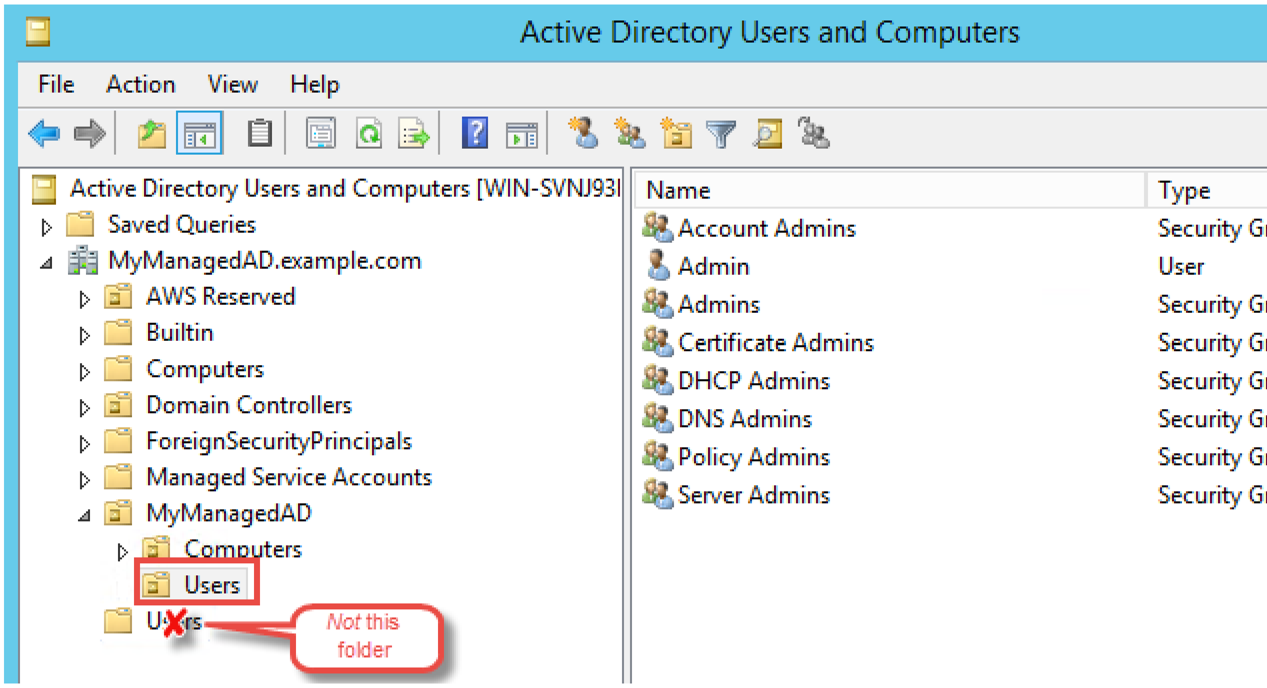
Type	Protocol	Port Range	Source
All traffic	All	All	The current security group (The security group for your directory).

Ensure That Kerberos Pre-authentication Is Enabled

Now you want to confirm that users in your Microsoft AD also have Kerberos pre-authentication enabled. This is the same process you completed for your on-premises directory. This is the default, but let's check to make sure nothing has changed.

To view user Kerberos settings

1. Log in to an instance that is a member of your Microsoft AD using an account that has domain administrative privileges.
2. If they are not already installed, install the Active Directory Users and Computers tool and the DNS tool. Learn how to install these tools in Installing the Active Directory Administration Tools (p. 148).
3. Open Server Manager. On the **Tools** menu, choose **Active Directory Users and Computers**.
4. Choose the **Users** folder in your domain. Note that this is the **Users** folder under your NetBIOS name, not the **Users** folder under the fully qualified domain name (FQDN).Open the context (right-click) menu for a user account and choose **Properties**.



Name	Type
Account Admins	Security Gr
Admin	User
Admins	Security Gr
Certificate Admins	Security Gr
DHCP Admins	Security Gr
DNS Admins	Security Gr
Policy Admins	Security Gr
Server Admins	Security Gr

5. Choose the **Account** tab. In the **Account options** list, ensure that **Do not require Kerberos preauthentication** is *not* checked.

Next Step

Step 3: Create the Trust Relationship

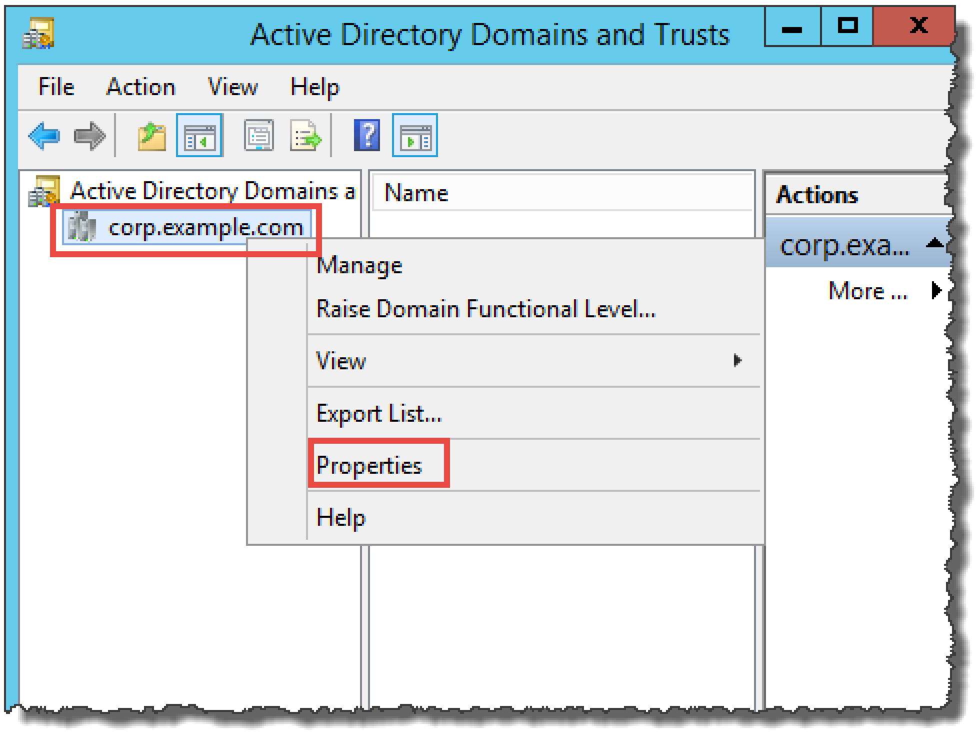
Now that the preparation work is complete, the final steps are to create the trusts. First you create the trust on your on-premises domain, and then finally on your Microsoft AD.

Configure Your On-Premises Trust

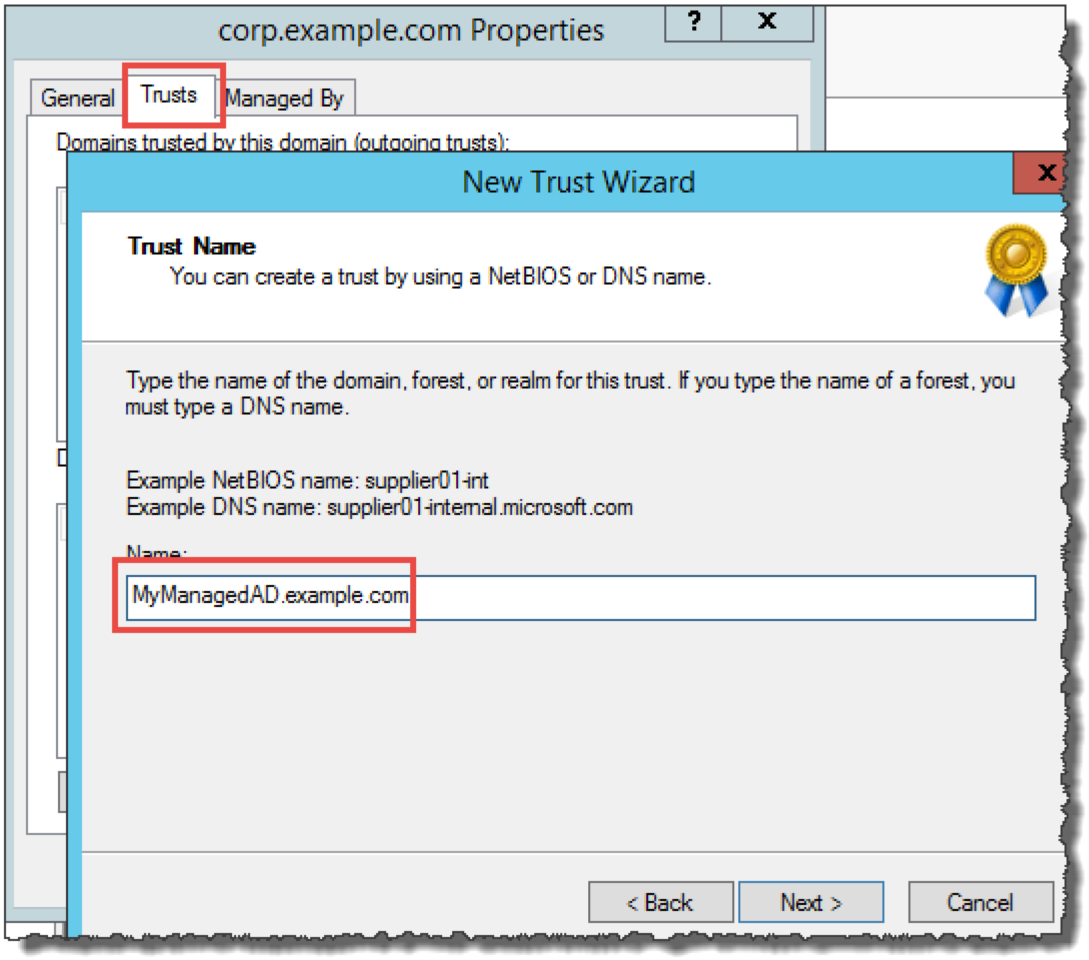
In this tutorial, you configure a two-way trust. However, if you create a one-way trust, be aware that the trust directions on each of your domains must be complementary. For example, if you create a one-way, outgoing trust on your on-premises domain, you need to create a one-way, incoming trust on your Microsoft AD.

To configure your on-premises trust relationship

1. Open Server Manager and on the **Tools** menu, choose **Active Directory Domains and Trusts**.
2. Open the context (right-click) menu of your domain and choose **Properties**.



3. Choose the **Trusts** tab and choose **New trust**. Type the name of your AWS Microsoft AD and choose **Next**.


corp.example.com Properties
General
Trusts
Managed By
Domains trusted by this domain (outgoing trusts):
New Trust Wizard

Trust Name

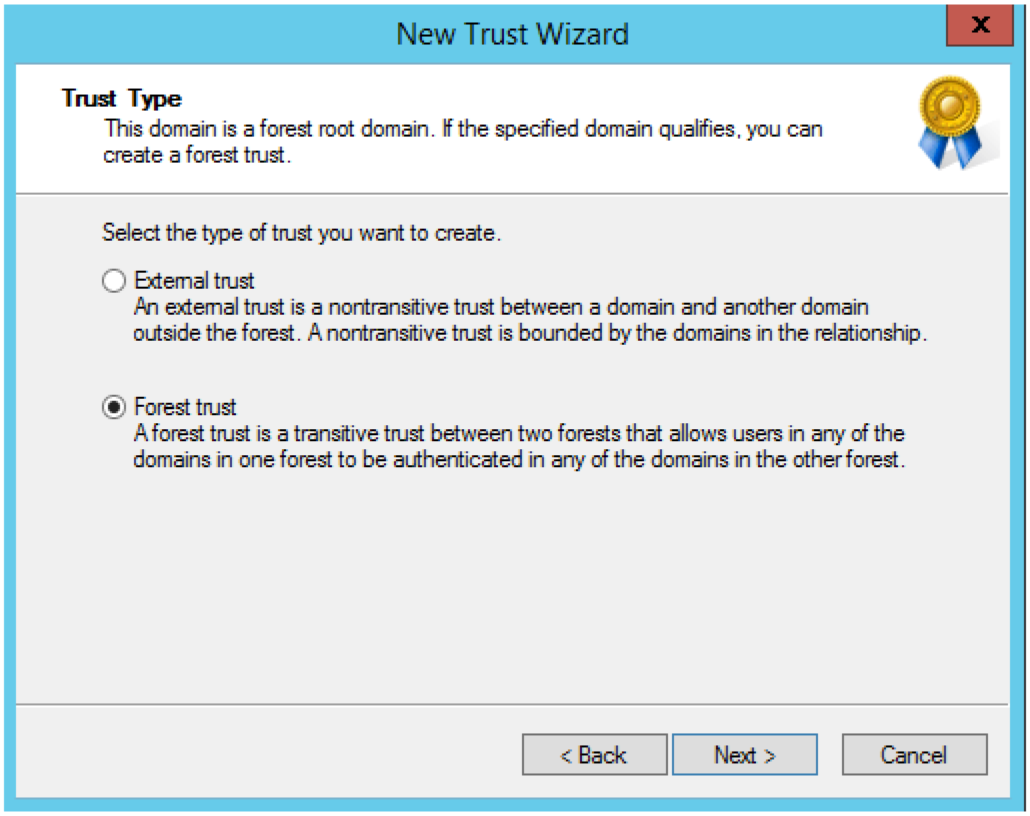
You can create a trust by using a NetBIOS or DNS name.

Type the name of the domain, forest, or realm for this trust. If you type the name of a forest, you must type a DNS name.

Example NetBIOS name: supplier01-int
Example DNS name: supplier01-internal.microsoft.com

Name:
MyManagedAD.example.com

4. Choose **Forest trust**. Choose **Next**.



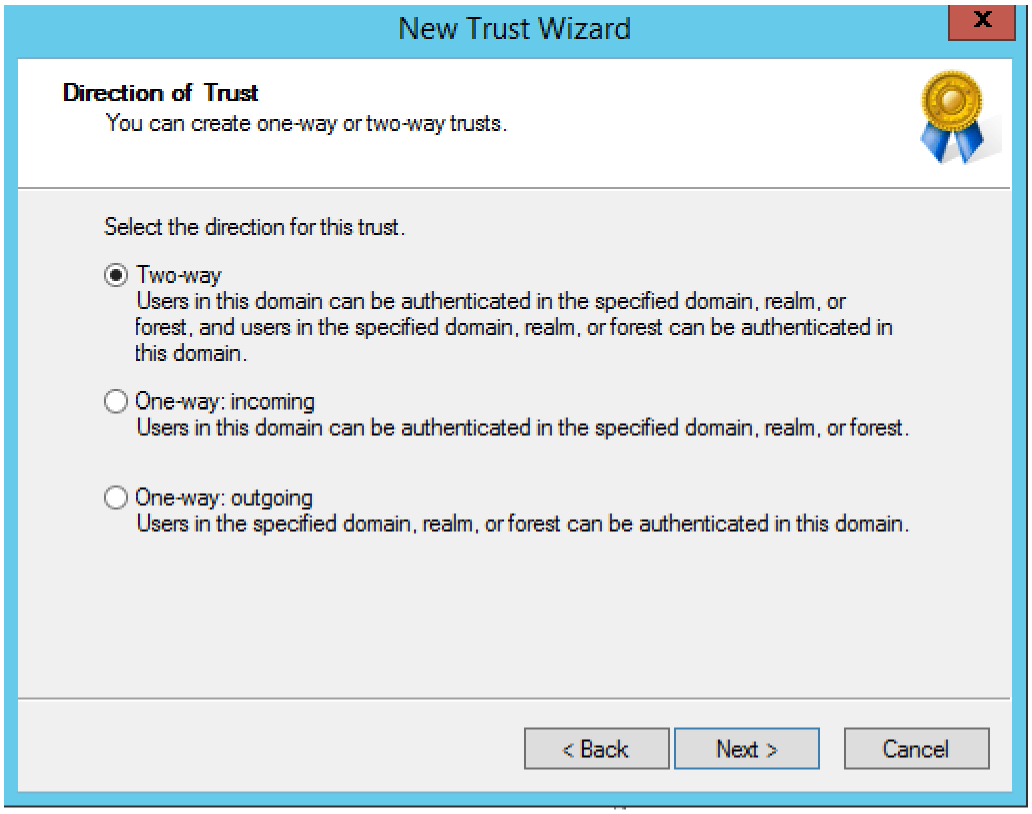
Trust Type

This domain is a forest root domain. If the specified domain qualifies, you can create a forest trust.

Select the type of trust you want to create.

- ○ External trust
  An external trust is a nontransitive trust between a domain and another domain outside the forest. A nontransitive trust is bounded by the domains in the relationship.
- ● Forest trust
  A forest trust is a transitive trust between two forests that allows users in any of the domains in one forest to be authenticated in any of the domains in the other forest.

5. Choose **Two-way**. Choose **Next**.



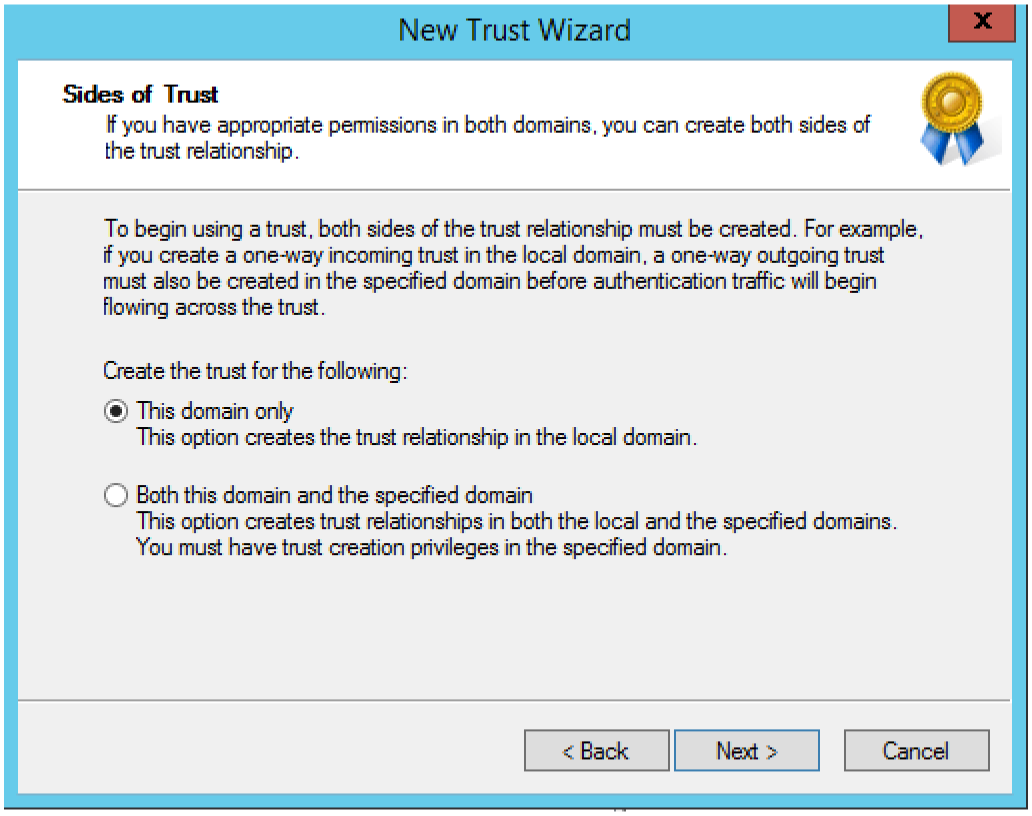
Direction of Trust

You can create one-way or two-way trusts.

Select the direction for this trust.

- ● Two-way
  Users in this domain can be authenticated in the specified domain, realm, or forest, and users in the specified domain, realm, or forest can be authenticated in this domain.
- ○ One-way: incoming
  Users in this domain can be authenticated in the specified domain, realm, or forest.
- ○ One-way: outgoing
  Users in the specified domain, realm, or forest can be authenticated in this domain.

6. Choose **This domain only**. Choose **Next**.



Sides of Trust

If you have appropriate permissions in both domains, you can create both sides of the trust relationship.

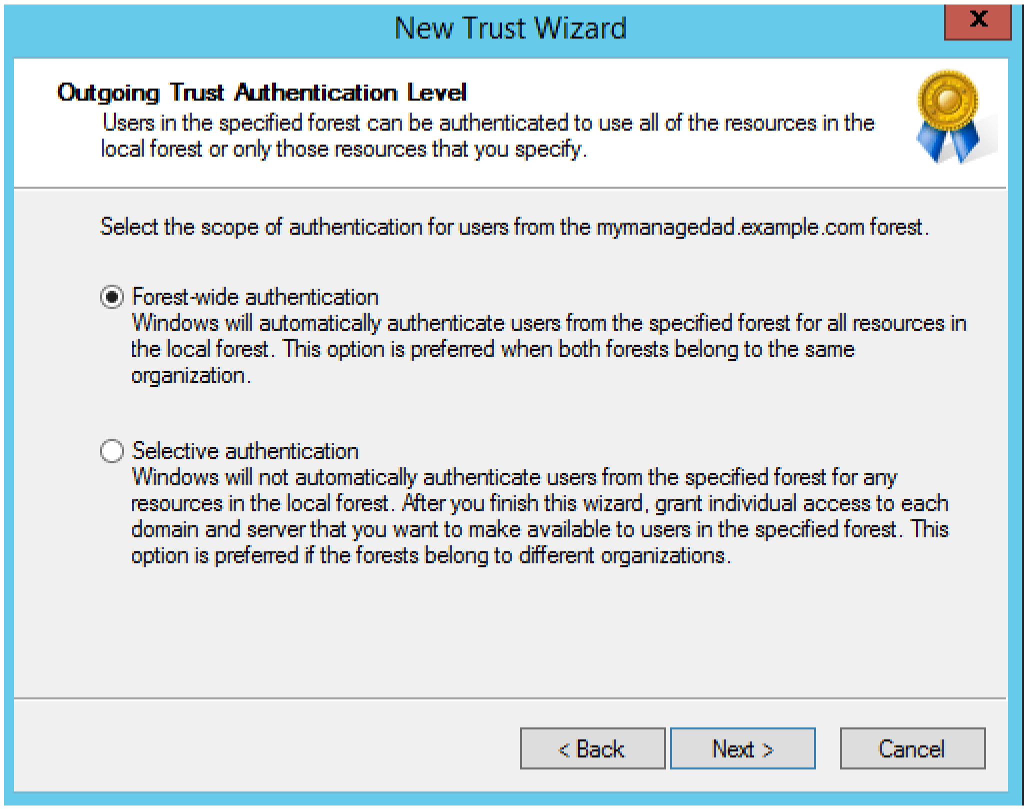
To begin using a trust, both sides of the trust relationship must be created. For example, if you create a one-way incoming trust in the local domain, a one-way outgoing trust must also be created in the specified domain before authentication traffic will begin flowing across the trust.

Create the trust for the following:

● This domain only
This option creates the trust relationship in the local domain.

○ Both this domain and the specified domain
This option creates trust relationships in both the local and the specified domains. You must have trust creation privileges in the specified domain.

7. Choose **Forest-wide authentication**. Choose **Next**.



Outgoing Trust Authentication Level

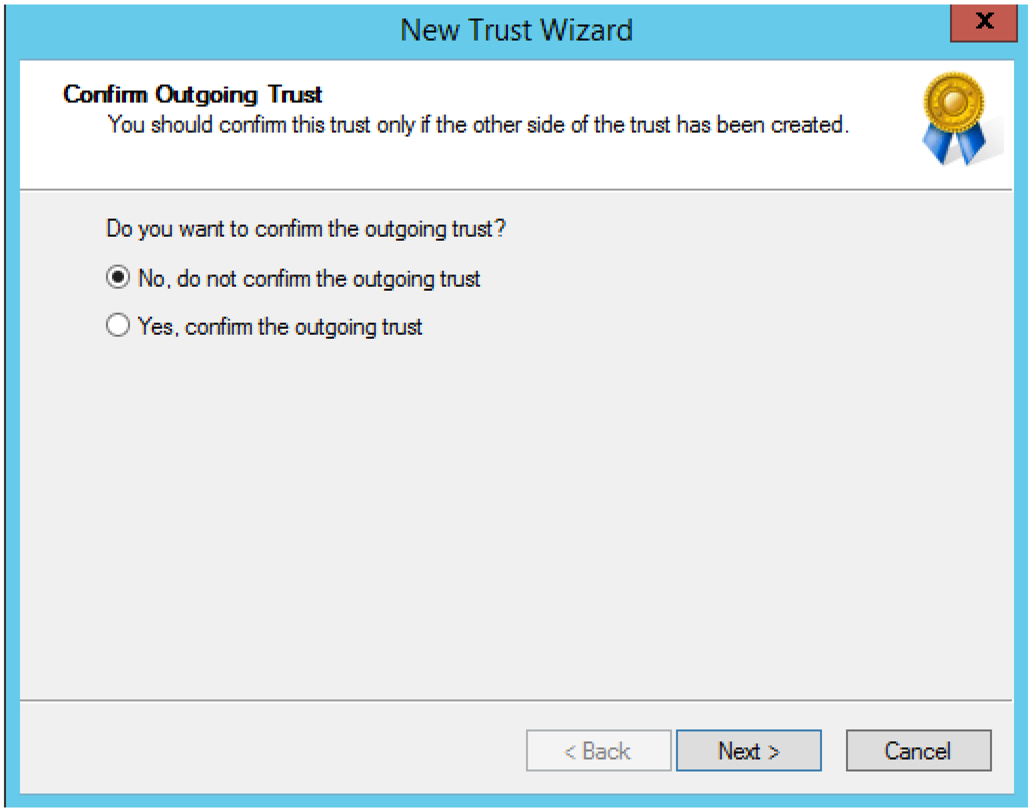
Users in the specified forest can be authenticated to use all of the resources in the local forest or only those resources that you specify.

Select the scope of authentication for users from the mymanagedad.example.com forest.

● Forest-wide authentication
Windows will automatically authenticate users from the specified forest for all resources in the local forest. This option is preferred when both forests belong to the same organization.

○ Selective authentication
Windows will not automatically authenticate users from the specified forest for any resources in the local forest. After you finish this wizard, grant individual access to each domain and server that you want to make available to users in the specified forest. This option is preferred if the forests belong to different organizations.

8. Type a **Trust password**. Make sure to remember this password as you will need it when setting up the trust for your AWS Microsoft AD.
9. In the next dialog box, confirm your settings and choose **Next**. Confirm that the trust was created successfully and again choose **Next**.
10. Choose **No, do not confirm the outgoing trust**. Choose **Next**.



New Trust Wizard

Confirm Outgoing Trust

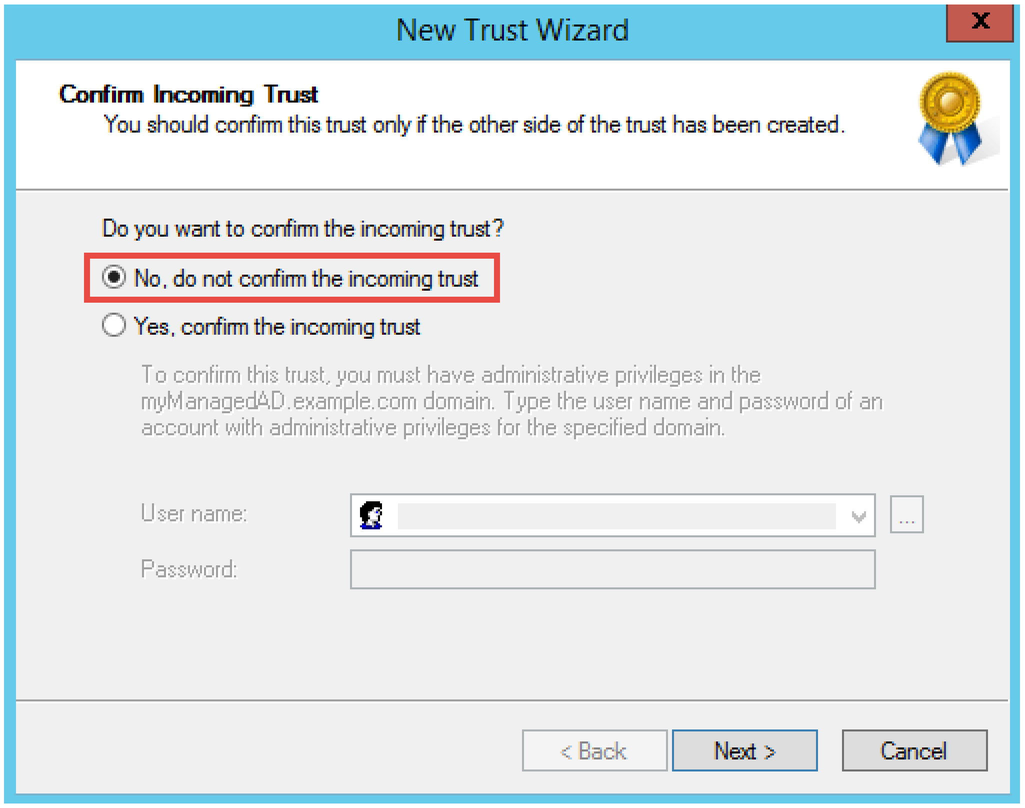
You should confirm this trust only if the other side of the trust has been created.

Do you want to confirm the outgoing trust?

[✓ No, do not confirm the outgoing trust]

Yes, confirm the outgoing trust

11. Choose **No, do not confirm the incoming trust**. Choose **Next**.



New Trust Wizard

Confirm Incoming Trust

You should confirm this trust only if the other side of the trust has been created.

Do you want to confirm the incoming trust?

[✓ No, do not confirm the incoming trust]

Yes, confirm the incoming trust

To confirm this trust, you must have administrative privileges in the myManagedAD.example.com domain. Type the user name and password of an account with administrative privileges for the specified domain.

User name:

Password:

Configure Your Microsoft AD Trust

Finally, you configure the trust relationship for your AWS Microsoft AD. Because you created a two-way trust on the on-premises domain, you also create a two-way trust on our Microsoft AD.

To configure your Microsoft AD trust relationship

1. Return to the AWS Directory Service console. On the **Directory Details** page, choose your Microsoft AD ID.
2. Choose the **Trust relationships** tab.
3. Choose **Add trust relationship**.
4. Type the FQDN of your on-premises domain (in this tutorial `corp.example.com`). Type the same trust password that you used when creating the trust on your on-premises domain. Specify the direction. In this case we choose **Two-way**.
5. In the **Conditional forwarder** field, enter the IP address of your on-premises DNS server. In this example, enter 172.16.10.153.
6. (Optional) Choose **Add IP address** and enter a second IP address your on-premises DNS server. You can specify up to a total of four DNS servers.
7. Choose **Create**.

Congratulations. You now have a trust relationship between your on-premises domain (corp.example.com) and your AWS Microsoft AD (MyManagedAD.example.com). Only one relationship can be set up between these two domains. If for example, you want to change the trust direction to one-way, you would first need to delete this existing trust relationship and create a new one.

For more information, including instructions about verifying or deleting trusts, see When to Create a Trust Relationship (p. 66).

Microsoft AD Test Lab Tutorials

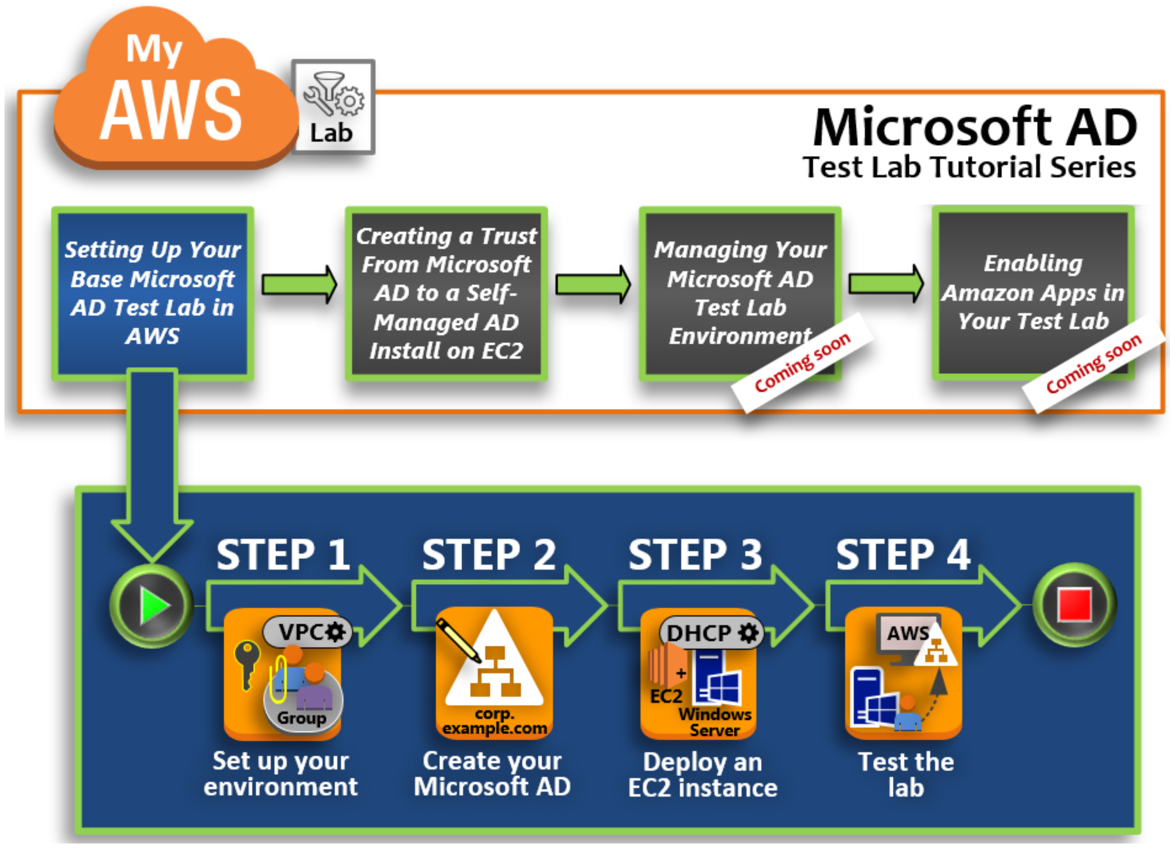
This section provides a series of guided tutorials to help you establish a test lab environment in AWS where you can experiment with Microsoft AD.

Topics

Tutorial: Setting Up Your Base Microsoft AD Test Lab in AWS

This tutorial teaches you how to set up your AWS environment to prepare for a new Microsoft AD installation that uses a new EC2 instance running Windows Server 2016. It then teaches you to use typical Active Directory administration tools to manage your Microsoft AD environment from your Windows system. By the time you complete the tutorial, you will have set up the network prerequisites and have configured a new Microsoft AD forest.

As shown in the following illustration, the lab you create from this tutorial is the foundational component for hands-on learning about Microsoft AD. You can later add optional tutorials for more hands-on experience. This tutorial series is ideal for anyone who is new to Microsoft AD and wants a test lab for evaluation purposes. This tutorial takes approximately 1 hour to complete.



Step 1: Set Up Your AWS Environment for Microsoft AD (p. 95)

After you've completed your prerequisite tasks, you create and configure a VPC in your EC2 instance.

Step 2: Create Your Microsoft AD Directory in AWS (p. 97)

In this step, you set up Microsoft AD in AWS for the first time.

Step 3: Deploy an EC2 Instance to Manage Microsoft AD (p. 98)

Here, you walk through the various post-deployment tasks necessary for client computers to connect to your new domain and set up a new Windows Server system in EC2.

Step 4: Verify That the Base Test Lab Is Operational (p. 101)

Finally, as an administrator, you verify that you can log in and connect to Microsoft AD from your Windows Server system in EC2. Once you've successfully tested that the lab is operational, you can continue to add other test lab guide modules.

Prerequisites

If you plan to use only the UI steps in this tutorial to create your test lab, you can skip this prerequisites section and move on to Step 1. However, if you plan to use either AWS CLI commands or AWS Tools for Windows PowerShell modules to create your test lab environment, you must first configure the following:

- **IAM user with the access and secret access key** – An IAM user with an access key is required if you want to use the AWS CLI or AWS Tools for Windows PowerShell modules. If you do not have an access key, see Creating, Modifying, and Viewing Access Keys (AWS Management Console).

- **AWS Command Line Interface (optional)** – Download and install the latest version of the AWS CLI for Windows in either 32 Bit or 64 bit. Once installed, open the command prompt or Windows PowerShell window, and then type **`aws configure`**. Note that you need the access key and secret key to complete the setup. See the first prerequisite for steps on how to do this. You will be prompted for the following:
  - AWS access key ID [None]: `AKIAIOSFODNN7EXAMPLE`
  - AWS secret access key [None]: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
  - Default region name [None]: `us-west-2`
  - Default output format [None]: `json`
- **AWS Tools for Windows PowerShell (optional)** – Download and install the latest version of the AWS Tools for Windows PowerShell from https://aws.amazon.com/powershell/, and then run the following command. Note that you need your access key and secret key to complete the setup. See the first prerequisite for the steps on how to do this.

  ```
  Set-AWSCredentials -AccessKey {AKIAIOSFODNN7EXAMPLE} -SecretKey
  {wJalrXUtnFEMI/K7MDENG/ bPxRfiCYEXAMPLEKEY} -StoreAs {default}
  ```

Step 1: Set Up Your AWS Environment for Microsoft AD

Before you can create Microsoft AD in your AWS test lab, you first need to set up your Amazon EC2 key pair so that all login data is encrypted.

Create a Key Pair

If you already have a key pair, you can skip this step. For more information about Amazon EC2 key pairs, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html.

To create a key pair

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Network & Security**, choose **Key Pairs**, and then choose **Create Key Pair**.
3. For **Key pair name**, type **`AWS-DS-KP`**, and then choose **Create**.
4. The private key file is automatically downloaded by your browser. The file name is the name you specified when you created your key pair with an extension of `.pem`. Save the private key file in a safe place.

   Important
   This is the only chance for you to save the private key file. You need to provide the name of your key pair when you launch an instance and the corresponding private key each time you decrypt the password for the instance.

Create a VPC

In this procedure you use a AWS CloudFormation template to create your VPC network. For more information about how this template works, see Amazon VPC QuickStart guide. For this tutorial, we will set up a public VPC. However you can make your VPC a private VPC as long as you create a network path to your VPC with Amazon Direct Connect, or with a Virtual Private Network (VPN) connection.

If you would like to use the default VPC (172.31.0.0/16), you can advance to the next section. All of the AWS CLI and PowerShell examples use custom VPC information. For more information, see Amazon Virtual Private Cloud (Amazon VPC).

To create a VPC

1. Launch the AWS CloudFormation template into your AWS account. Click here while signed on to your account.

   The template is launched in the US West (Oregon) Region by default. To change the region, use the region selector in the navigation bar. This stack takes approximately five minutes to create.
2. On the **Select Template** page, keep the default setting for the Amazon S3 template URL and then choose **Next**.
3. On the **Specify Details** page, set the stack name to **AWS-DS-VPC**. Then do the following:
   - Under **Availability Zone Configuration** select two zones in your area. Then for **Number of Availability Zones** select **2**.
   - Under **Network Configuration**, set **Create private subnets** to **false**. Then in **Create additional private subnets with dedicated network ACLs**, select **false**.
   - Under **Amazon EC2 Configuration**, set **Key pair name** to **`AWS-DS-KP`** or use an existing key pair.
4. Review your information, and then choose **Next**.
5. On the **Options** page, choose **Next**.
6. On the **Review** page, choose **Create**.

Create a Security Group for EC2 Instances

By default, Microsoft AD creates a security group to manage traffic between its domain controllers. In this procedure, you create a security group that manages traffic within your VPC for your EC2 instances. You also add a rule that allows RDP (3389) inbound from anywhere and for all traffic types inbound from the local VPC. For more information, see Amazon EC2 Security Groups for Windows Instances.

To create a security group for EC2 instances

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Under **Security**, choose **Security Groups**.
3. Choose **Create Security Group**.
4. In the **Create Security Group** dialog box provide the following values:
   - For **Name tag**, type **`AWS DS RDP Security Group`**.
   - For **Group name**, type **`AWS DS RDP Security Group`**.
   - For **Description**, type **`AWS DS RDP Security Group`**.
   - For **VPC**, select the VPC that ends with **AWS-DS-VPC**.
5. Choose **Yes, Create**.
6. Select **AWS DS RDP Security Group**.
7. Choose **Inbound Rules** below the list of security groups.
8. Choose **Edit**, and then choose **Add another rule**.
9. In the table, add the following values:
   - For **Type**, choose **RDP (3389)**.
   - For **Protocol**, verify that **TCP (6)** is displayed.
   - For **Port Range**, verify that **3389** is displayed.
   - For **Source**, specify a single IP address, or an IP address range in CIDR notation (for example, 203.0.113.5/32). You can also specify the name or ID of another security group in the same region. This setting determines the traffic that can reach your EC2 instances. For more information, see Understand Your Directory's AWS Security Group Configuration and Use (p. 8).
10. Choose **Add another rule**, and then provide the following values:

- For **Type**, choose **All Traffic**.
- For **Protocol**, verify that **All** is displayed.
- For **Port Range**, verify that **All** is displayed.
- For **Source**, type **10.0.0.0/16**.

11. Choose **Save**.

Step 2: Create Your Microsoft AD Directory in AWS

You can use three different methods to create your directory. You can use the AWS Management Console procedure (recommended for this tutorial) or you can use either the AWS CLI or AWS Tools for Windows PowerShell procedures to create your directory.

Method 1: To create your Microsoft AD directory (AWS Management Console)

1. Open the AWS Directory Service console at https://console.aws.amazon.com/directoryservice/.
2. In the AWS Directory Service console navigation pane, choose **Directories**.
3. Choose **Set up directory**, and then choose **Microsoft AD**.
4. On the **Directory details** page, provide the following information:
   - For **Edition**, select either **Standard** or **Enterprise** edition. For more information about editions, see AWS Directory Service for Microsoft Active Directory (p. 1).
   - For **Directory DNS**, type **`corp.example.com`**.
   - For **NetBIOS name**, type **`corp`**.
   - For **Admin password**, type the password you want to use for this account and type the password again in **Confirm password**. This **Admin** account is automatically created during the directory creation process. The password cannot include the word *admin*. The directory administrator password is case sensitive and must be between 8 and 64 characters in length, inclusive. It must also contain at least one character from three of the following four categories:
     - Lowercase letters (a-z)
     - Uppercase letters (A-Z)
     - Numbers (0-9)
     - Non-alphanumeric characters (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)
   - For **Description**, type **`AWS DS Managed`**.
   - For **VPC**, choose the option that begins with **AWS-DS-VPC** and ends with **(10.0.0.0/16)**.
   - For **Subnets**, choose the **10.0.128.0/20** and **10.0.144.0/20** public subnets.
5. Choose **Next Step**.
6. Review the directory information and make any necessary changes. If the information is correct, choose **Create Microsoft AD**.
7. Creating the directory takes 20 to 40 minutes. Once created, the **Status** value changes to **Active**.
8. Choose **Done**.

Method 2: To create your Microsoft AD (Windows PowerShell) (Optional)

1. Open Windows PowerShell.
2. Type the following command. Make sure to use the values provided in Step 4 of the preceding AWS Management Console procedure.

```
New-DSMicrosoftAD -Name corp.example.com -ShortName corp -Password
P@ssw0rd -Description “AWS DS Managed” - VpcSettings_VpcId vpc-xxxxxxxx -
VpcSettings_SubnetId subnet-xxxxxxxx, subnet-xxxxxxxx
```

Method 3: To create your Microsoft AD (AWS CLI) (Optional)

1. Open the AWS CLI.
2. Type the following command. Make sure to use the values provided in Step 4 of the preceding AWS Management Console procedure.

```
aws create-microsoft-ad --name corp.example.com --short-name corp --
password P@ssw0rd --description "AWS DS Managed" --vpc-settings VpcId= vpc-
xxxxxxxx,SubnetIds= subnet-xxxxxxxx, subnet-xxxxxxxx
```

Step 3: Deploy an EC2 Instance to Manage Microsoft AD

For this lab, we are using EC2 instances that have public IP addresses to make it easy to access the management instance from anywhere. In a production setting, you can use instances that are in a private VPC that are only accessible through a VPN or Amazon Direct Connect link. There is no requirement the instance have a public IP address.

In this section, you walk through the various post-deployment tasks necessary for client computers to connect to your domain using the Windows Server on your new EC2 instance. You use the Windows Server in the next step to verify that the lab is operational.

Create a DHCP Options Set for Your Directory

In this procedure, you set up a DHCP option scope so that EC2 instances in your VPC automatically use your Microsoft AD for DNS resolution. For more information, see DHCP Options Sets.

To create a DHCP options set for your directory

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose **DHCP Options Sets** in the navigation pane. Then choose **Create DHCP options set**.
3. In the **Create DHCP options set** dialog box, provide the following values for your directory:
   - For **Name tag**, type **`AWS DS DHCP`**.
   - For **Domain name**, type **`corp.example.com`**.
   - For **Domain name servers**, type the IP addresses of your AWS provided directory's DNS servers. To find these addresses, go to the AWS Directory Service console navigation pane, choose **Directories**, choose the applicable directory ID, and on the **Details** page use the IPs that are displayed in **DNS address**.
   - Leave the settings blank for **NTP servers**, **NetBIOS name servers**, and **NetBIOS node type**.
4. Choose **Yes, Create**. The new set of DHCP options appear in your list of DHCP options.
5. Make a note of the ID of the new set of DHCP options (**dopt-`xxxxxxxx`**). You need it at the end of this procedure when you associate the new options set with your VPC.
6. Choose **Your VPCs** in the navigation pane.
7. Select the **AWS DS DHCP** VPC, choose **Actions**, and then choose **Edit DHCP Options Set**.
8. In the **Edit DHCP Options Set** dialog box, select the options set that you recorded in Step 5. Then choose **Save**.

Create a Role to Join Windows Instances to Your Microsoft AD Domain

Use this procedure to configure a role that joins an EC2 Windows instance to a domain. For more information, see Joining a Windows Instance to an AWS Directory Service Domain.

To configure EC2 to join Windows instances to your domain

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. Under **AWS service**, select the **EC2** link, and then click **Next**.
4. Under **Select your use case**, select **EC2**, and then select **Next**.
5. In the list of polices, select the **AmazonEC2RoleforSSM** policy, and then select **Next**.
6. In **Role name**, type a name for the role that describes that it is used for domain join. In this case use **EC2DomainJoin**, and then select the **Create role** button.

Create an EC2 Instance and Automatically Join the Directory

In this procedure you set up a Windows Server system in Amazon EC2 that can be used later to administer users, groups, and policies in Active Directory.

To create an EC2 instance and automatically join the directory

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Choose **Launch Instance**.
3. On the **Step 1** page, next to **Microsoft Windows Server 2016 Base - ami-*`xxxxxxxxx`*** choose **Select**.
4. On the **Step 2** page, select **t2.micro** (note, you can choose a larger instance type), and then choose **Next: Configure Instance Details**.
5. On the **Step 3** page, do the following:
   - For **Network**, choose the VPC that ends with **AWS-DS-VPC** (for example, **vpc-*`xxxxxxxxx`* | AWS-DS-VPC**).
   - For **Subnet** choose **Public subnet 1**, which should be preconfigured for your preferred Availability Zone (for example, **subnet-*`xxxxxxxxx`* | Public subnet1 |** *`us-west-2a`*).
   - For **Auto-assign Public IP**, choose **Enable** (if the subnet setting is not set to enable by default).
   - For **Domain join directory**, choose **corp.example.com (d-*`xxxxxxxxxxx`*)**.
   - For **IAM role** choose **EC2DomainJoin**.
   - Leave the rest of the settings at their defaults.
   - Choose **Next: Add Storage**.
6. On the **Step 4** page, leave the default settings, and then choose **Next: Add Tags**.
7. On the **Step 5** page, choose **Add Tag**. Under **Key** type **`corp.example.com-mgmt`** and then choose **Next: Configure Security Group**.
8. On the **Step 6** page, choose **Select an existing security group**, select **AWS DS RDP Security Group**, and then choose **Review and Launch** to review your instance.
9. On the **Step 7** page, review the page, and then choose **Launch**.
10. On the **Select an existing key pair or create a new key pair** dialog box, do the following:
    - Choose **Choose an existing key pair**.
    - Under **Select a key pair**, choose **AWS-DS-KP**.
    - Select the **I acknowledge...** check box.
    - Choose **Launch Instances**.
11. Choose **View Instances** to return to the Amazon EC2 console and view the status of the deployment.

Install the Active Directory Tools on Your EC2 Instance

You can choose from two methods to install the Active Directory Domain Management Tools on your EC2 instance. You can use the Server Manager UI (recommended for this tutorial) or Windows PowerShell.

To install the Active Directory Tools on your EC2 instance (Server Manager)

1. In the Amazon EC2 Console, choose **Instances**, select the instance you just created, and then choose **Connect**.
2. In the **Connect To Your Instance** dialog box, choose **Download Remote Desktop File**.
3. In the **Windows Security** dialog box, type your local administrator credentials for the Windows Server computer to log in (for example, **`administrator`**).
4. From the **Start** menu, choose **Server Manager**.
5. In the **Dashboard**, choose **Add Roles and Features**.
6. In the **Add Roles and Features Wizard**, choose **Next**.
7. On the **Select installation type** page, choose **Role-based or feature-based installation**, and then choose **Next**.
8. On the **Select destination server** page, make sure that the local server is selected, and then choose **Next**.
9. On the **Select server roles** page, choose **Next**.
10. On the **Select features** page, do the following:
    - Select the **Group Policy Management** check box.
    - Expand **Remote Server Administration Tools**, and then expand **Role Administration Tools**.
    - Select the **AD DS and AD LDS Tools** check box.
    - Select the **DNS Server Tools** check box.
    - Choose **Next**.
11. On the **Confirm installation selections** page, review the information, and then choose **Install**. When the feature installation is finished, the following new tools or snap-ins will be available in the Windows Administrative Tools folder in the Start menu.
    - Active Directory Administrative Center
    - Active Directory Domains and Trusts
    - Active Directory Module for Windows PowerShell
    - Active Directory Sites and Services
    - Active Directory Users and Computers
    - ADSI Edit
    - DNS
    - Group Policy Management

To install the Active Directory Tools on your EC2 instance (Windows PowerShell) (Optional)

1. Start Windows PowerShell.
2. Type the following command.

```
Install-WindowsFeature -Name GPMC,RSAT-AD-PowerShell,RSAT-AD-
AdminCenter,RSAT-ADDS-Tools,RSAT-DNS-Server
```

Step 4: Verify That the Base Test Lab Is Operational

Use the following procedure to verify that the test lab has been set up successfully before adding on additional test lab guide modules. This procedure verifies that your Windows Server is configured appropriately, can connect to the corp.example.com domain, and be used to administer your Microsoft AD forest.

To verify that the test lab is operational

1. Sign out of the EC2 instance where you were logged in as the local administrator.
2. Back in the Amazon EC2 console, choose **Instances** in the navigation pane. Then select the instance that you created. Choose **Connect**.
3. In the **Connect To Your Instance** dialog box, choose **Download Remote Desktop File**.
4. In the **Windows Security** dialog box, type your administrator credentials for the CORP domain to log in (for example, **`corp\admin`**).
5. Once you are logged in, in the **Start** menu, under **Windows Administrative Tools**, choose **Active Directory Users and Computers**.
6. You should see **corp.example.com** displayed with all the default OUs and accounts associated with a new domain. Under **Domain Controllers**, notice the names of the domain controllers that were automatically created when you created your Microsoft AD in AWS back in Step 2 of this tutorial.

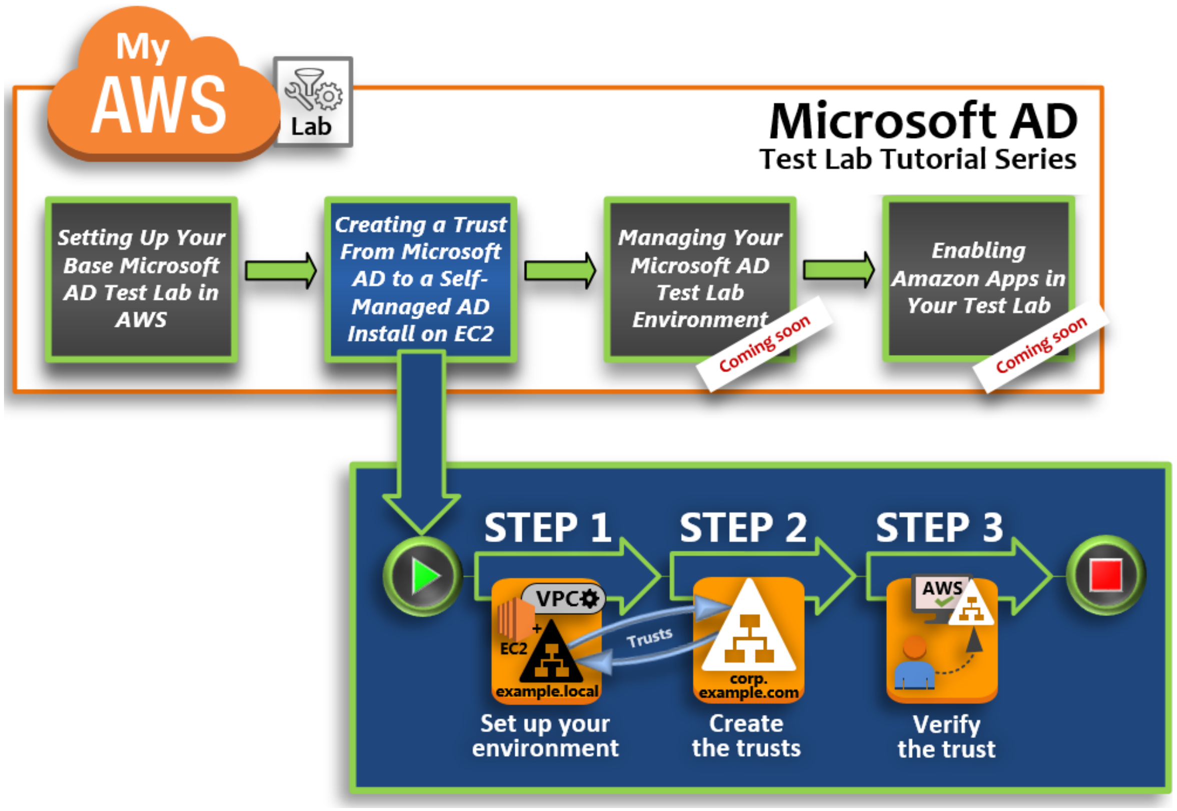
Congratulations! Your Microsoft AD base test lab environment has now been configured. You are ready to begin adding the next test lab in the series.

Next tutorial: Tutorial: Creating a Trust from Microsoft AD to a Self-Managed Active Directory Installation on Amazon EC2 (p. 101)

Tutorial: Creating a Trust from Microsoft AD to a Self-Managed Active Directory Installation on Amazon EC2

In this tutorial, you learn how to create a trust between the AWS Directory Service for Microsoft Active Directory forest you that created in the Base tutorial (p. 93). You also learn to create a new native Active Directory forest on a Windows Server in Amazon EC2. As shown in the following illustration, the lab that you create from this tutorial is the second building block necessary when setting up a complete Microsoft AD test lab. You can use the test lab to test your pure cloud or hybrid cloud–based AWS solutions.

You should only need to create this tutorial once. After that you can add optional tutorials when necessary for more experience.


My AWS
Lab
Microsoft AD
Test Lab Tutorial Series
Setting Up Your Base Microsoft AD Test Lab in AWS
Creating a Trust From Microsoft AD to a Self-Managed AD Install on EC2
Managing Your Microsoft AD Test Lab Environment
Coming soon
Enabling Amazon Apps in Your Test Lab
Coming soon
STEP 1
STEP 2
STEP 3
VPC
EC2
example.local
Trusts
corp.example.com
AWS
Set up your environment
Create the trusts
Verify the trust

Step 1: Set Up Your Environment for Trusts (p. 102)

Before you can establish trusts between a new Active Directory forest and the Microsoft AD forest that you created in the Base tutorial (p. 93), you need to prepare your Amazon EC2 environment. To do that, you first create a Windows Server 2016 server, promote that server to a domain controller, and then configure your VPC accordingly.

Step 2: Create the Trusts (p. 105)

In this step, you create a two-way forest trust relationship between your newly created Active Directory forest hosted in Amazon EC2 and your Microsoft AD forest in AWS.

Step 3: Verify the Trust (p. 106)

Finally, as an administrator, you use the AWS Directory Service console to verify that the new trusts are operational.

Step 1: Set Up Your Environment for Trusts

In this section, you set up your Amazon EC2 environment, deploy your new forest, and prepare your VPC for trusts with AWS.

Create a Windows Server 2016 EC2 Instance

Use the following procedure to create a Windows Server 2016 member server in Amazon EC2.

To create a Windows Server 2016 EC2 instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. In the Amazon EC2 console, choose **Launch Instance**.
3. On the **Step 1** page, locate **Microsoft Windows Server 2016 Base - ami-*xxxxxxxx*** in the list. Then choose **Select**.
4. On the **Step 2** page, select **t2.large**, and then choose **Next: Configure Instance Details**.
5. On the **Step 3** page, do the following:

    - For **Network**, select **vpc-*xxxxxxxx* AWS-DS-VPC** (which you previously set up in the Base tutorial (p. 95)).
    - For **Subnet**, select **subnet-*xxxxxxxx* | Public subnet2 | (YourAZ)**.
    - For **Auto-assign Public IP** list, choose **Enable** (if the subnet setting is not set to **Enable** by default).
    - Leave the rest of the settings at their defaults.
    - Choose **Next: Add Storage**.
6. On the **Step 4** page, leave the default settings, and then choose **Next: Add Tags**.
7. On the **Step 5** page, choose **Add Tag**. Under **Key** type **`example.local-DC01`**, and then choose **Next: Configure Security Group**.
8. On the **Step 6** page, choose **Select an existing security group**, select **AWS DS RDP Security Group** (which you previously set up in the Base tutorial (p. 96)), and then choose **Review and Launch** to review your instance.
9. On the **Step 7** page, review the page, and then choose **Launch**.
10. On the **Select an existing key pair or create a new key pair** dialog box, do the following:

    - Choose **Choose an existing key pair**.
    - Under **Select a key pair**, choose **AWS-DS-KP** (which you previously set up in the Base tutorial (p. 95)).
    - Select the **I acknowledge...** check box.
    - Choose **Launch Instances**.
11. Choose **View Instances** to return to the Amazon EC2 console and view the status of the deployment.

Promote Your Server to a Domain Controller

Before you can create trusts, you must build and deploy the first domain controller for a new forest. During this process you configure a new Active Directory forest, install DNS, and set this server to use the local DNS server for name resolution. You must reboot the server at the end of this procedure.

Note
If you want to create a domain controller in AWS that replicates with your on-premises network, you would first manually join the EC2 instance to your on-premises domain. After that you can promote the server to a domain controller.

To promote your server to a domain controller

1. In the Amazon EC2 console, choose **Instances**, select the instance you just created, and then choose **Connect**.
2. In the **Connect To Your Instance** dialog box, choose **Download Remote Desktop File**.
3. In the **Windows Security** dialog box, type your local administrator credentials for the Windows Server computer to login (for example, **`administrator`**). If you do not yet have the local administrator password, go back to the Amazon EC2 console, right-click on the instance, and choose **Get Windows Password**. Navigate to your `AWS DS KP.pem` file or your personal `.pem` key, and then choose **Decrypt Password**.
4. From the **Start** menu, choose **Server Manager**.
5. In the **Dashboard**, choose **Add Roles and Features**.

6. In the **Add Roles and Features Wizard**, choose **Next**.
7. On the **Select installation type** page, choose **Role-based or feature-based installation**, and then choose **Next**.
8. On the **Select destination server** page, make sure that the local server is selected, and then choose **Next**.
9. On the **Select server roles** page, select **Active Directory Domain Services**. In the **Add Roles and Features Wizard** dialog box, verify that the **Include management tools (if applicable)** check box is selected. Choose **Add Features**, and then choose **Next**.
10. On the **Select features** page, choose **Next**.
11. On the **Active Directory Domain Services** page, choose **Next**.
12. On the **Confirm installation selections** page, choose **Install**.
13. Once the Active Directory binaries are installed, choose **Close**.
14. When Server Manager opens, look for a flag at the top next to the word **Manage**. When this flag turns yellow, the server is ready to be promoted.
15. Choose the yellow flag, and then choose **Promote this server to a domain controller**.
16. On the **Deployment Configuration** page, choose **Add a new forest**. In **Root domain name**, type **`example.local`**, and then choose **Next**.
17. On the **Domain Controller Options** page, do the following:
    - In both **Forest functional level** and **Domain functional level**, choose **Windows Server 2016**.
    - Under **Specify domain controller capabilities**, verify that both **Domain Name System (DNS) server** and **Global Catalog (GC)** are selected.
    - Type and then confirm a Directory Services Restore Mode (DSRM) password. Then choose **Next**.
18. On the **DNS Options** page, ignore the warning about delegation and choose **Next**.
19. On the **Additional options** page, make sure that **EXAMPLE1** is listed as the NetBios domain name.
20. On the **Paths** page, leave the defaults, and then choose **Next**.
21. On **Review Options** page, choose **Next**. The server now checks to make sure all the prerequisites for the domain controller are satisfied. You may see some warnings displayed, but you can safely ignore them.
22. Choose **Install**. Once the installation is complete, the server reboots and then becomes a functional domain controller.

Configure Your VPC

The following three procedures guide you through the steps to configure your VPC for connectivity with AWS.

To configure your VPC outbound rules

1. In the AWS Directory Service console, make a note of the Microsoft AD directory ID for corp.example.com that you previously created in the Base tutorial (p. 97).
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. In the navigation pane, choose **Security Groups**.
4. Search for your Microsoft AD directory ID. In the search results, select the item with the description **AWS created security group for d-*`xxxxxx`***.

    Note
    This security group was automatically created when you initially created your directory.
5. Choose the **Outbound Rules** tab under that security group. Choose **Edit**, choose **Add another rule**, and then add the following values:

- For **Type**, choose **All Traffic**.
- For **Destination**, type `0.0.0.0/0`.
- Leave the rest of the settings at their defaults.
- Select **Save**.

To verify Kerberos preauthentication is enabled

1. On the **example.local** domain controller, open **Server Manager**.
2. On the **Tools** menu, choose **Active Directory Users and Computers**.
3. Navigate to the **Users** directory, right-click on any user account listed in the right pane, and then choose the **Account** tab. In the **Account options** list, scroll down and ensure that **Do not require Kerberos preauthentication** is **not** selected.
4. Perform the same steps for the **corp.example.com** domain from the **corp.example.com-mgmt** instance.

To configure DNS conditional forwarders

1. First you must get some information about your AWS Microsoft AD.

   Sign in to the AWS Management Console and open the AWS Directory Service console at https://console.aws.amazon.com/directoryservice/.
2. In the navigation pane, choose **Directories**.
3. Select the **directory ID** of your Microsoft AD.
4. Take note of the fully qualified domain name (FQDN), **corp.example.com**, and the DNS addresses of your directory.
5. Now, return to your **example.local** domain controller, and then open **Server Manager**.
6. On the **Tools** menu, choose **DNS**.
7. In the console tree, expand the DNS server of the domain for which you are setting up the trust, and navigate to **Conditional Forwarders**.
8. Right-click **Conditional Forwarders**, and then choose **New Conditional Forwarder**.
9. In DNS domain, type `corp.example.com`.
10. Under **IP addresses of the master servers**, choose **<Click here to add ...>**, type the first DNS address of your Microsoft AD directory (which you made note of in the previous procedure), and then press **Enter**. Do the same for the second DNS address. After typing the DNS addresses, you might get a "timeout" or "unable to resolve" error. You can generally ignore these errors.
11. Select the **Store this conditional forwarder in Active Directory, and replicate as follows** check box. In the drop-down menu, choose **All DNS servers in this Forest**, and then choose **OK**.

Step 2: Create the Trusts

In this section, you create two separate forest trusts. One trust is created from the Active Directory domain on your EC2 instance and the other from your Microsoft AD in AWS.

Note
The AWS Directory Service, currently only supports forest trusts.

To create the trust from your EC2 domain to your Microsoft AD in AWS

1. Log into **example.local**.

2. Open **Server Manager** and in the console tree choose **DNS**. Take note of the IPv4 address listed for the server. You will need this in the next procedure when you create a conditional forwarder from **corp.example.com** to the **example.local** directory.
3. In the **Tools** menu, choose **Active Directory Domains and Trusts**.
4. In the console tree, right-click **example.local** and then choose **Properties**.
5. On the **Trusts** tab, choose **New Trust**, and then choose **Next**.
6. On the **Trust Name** page, type **`corp.example.com`**, and then choose **Next**.
7. On the **Trust Type** page, choose **Forest trust**, and then choose **Next**.
8. On the **Direction of Trust** page, choose **Two-way**, and then choose **Next**.
9. On the **Sides of Trust** page, choose **This domain only**, and then choose **Next**.
10. On the **Outgoing Trust Authentication Level** page, choose **Forest-wide authentication**, and then choose **Next**.
11. On the **Trust Password** page, type the trust password twice, and then choose **Next**. You will use this same password in the next procedure.
12. On the **Trust Selections Complete** page, review the results, and then choose **Next**.
13. On the **Trust Creation Complete** page, review the results, and then choose **Next**.
14. On the **Confirm Outgoing Trust** page, choose **No, do not confirm the outgoing trust**. Then choose **Next**
15. On the **Confirm Incoming Trust** page, choose **No, do not confirm the incoming trust**. Then choose **Next**
16. On the **Completing the New Trust Wizard** page, choose **Finish**.

To create the trust from your Microsoft AD in AWS to your EC2 domain

1. Open the AWS Directory Service console.
2. Choose the **corp.example.com** directory.
3. On the **Details** page, choose the **Trust relationships** tab.
4. Choose **Add trust relationship**.
5. In the **Add a trust relationship** dialog box, do the following:
   - For **Remote domain name**, type **example.local**.
   - For **Trust password**, type the same password that you provided in the previous procedure.
   - In **Trust direction**, select **Two-way**.
   - In **Conditional forwarder**, type the IP address of your DNS server in the **example.local** forest (which you noted in the previous procedure).
6. Choose **Add**.

Step 3: Verify the Trust

In this section, you test whether the trusts were set up successfully between AWS and Active Directory on Amazon EC2.

To verify the trust

1. Open the AWS Directory Service console.
2. Choose the **corp.example.com** directory.
3. On the **Details** page, choose the **Trust relationships** tab.

4. Select the trust relationship you just created.
5. Choose **Actions**, and then choose **Verify trust relationship**.

Once the verification has completed, you should see **Verified** displayed under the **Status** column.

Congratulations on completing this tutorial! You now have a fully functional multiforest Active Directory environment from which you can begin testing various scenarios. Additional test lab tutorials are planned in 2017, so check back on occasion to see what's new.

Security

This section describes considerations for securing your Microsoft AD environment.

Topics

Secure LDAP Communications

Lightweight Directory Access Protocol (LDAP) is a standard communications protocol used to read and write data to and from Active Directory. Some applications use LDAP to add, remove, or search users and groups in Active Directory or to transport credentials for authenticating users in Active Directory.

By default, communications over LDAP are not encrypted. This makes it possible for a malicious user to use network monitoring software to view data packets over the wire. This is why many corporate security policies typically require that organizations encrypt all LDAP communication.

To mitigate this form of data exposure, AWS Microsoft AD provides an option for you to enable LDAP over Secure Sockets Layer (SSL)/Transport Layer Security (TLS), also known as LDAPS. With LDAPS, you can improve security across the wire and meet compliance requirements by encrypting all communications between your LDAP-enabled applications and AWS Microsoft AD directory.

Enable LDAPS

You must do most of the setup from the Amazon EC2 instance that you use to manage your AWS Microsoft AD domain controllers. The following steps guide you through enabling LDAPS for your domain in the AWS Cloud.

Topics

Step 1: Delegate Who Can Enable LDAPS

To enable LDAPS, you must be a member of the Admins or AWS Delegated Enterprise Certificate Authority Administrators group in your AWS Microsoft AD directory or be the default administrative user

(Admin account). If you prefer to have a user other than the Admin account setup LDAPS, then add this user to the Admins or AWS Delegated Enterprise Certificate Authority Administrators group in your AWS Microsoft AD directory.

Step 2: Set Up Your Certificate Authority

Before you can enable LDAPS, you must create a certificate issued by a Microsoft Enterprise Certificate Authority (CA) server that is joined to your AWS Microsoft AD domain. Once created, the certificate must be installed on each of your domain controllers in that domain. This certificate lets the LDAP service on the domain controllers listen for and automatically accept SSL connections from LDAP clients.

> **Note**
> LDAPS with AWS Microsoft AD does not support certificates that are issued by a standalone CA.

Depending on your business need, you have the following options for setting up or connecting to a CA in your domain:

- **Create a subordinate Microsoft Enterprise CA** – (Recommended) With this option you can deploy a subordinate Microsoft Enterprise CA server in the AWS Cloud that uses EC2 so that it works with your existing root Microsoft CA. For more information about how to set up a subordinate Microsoft Enterprise CA, see Install a Subordinate Certification Authority on the Microsoft TechNet website.
- **Create a root Microsoft Enterprise CA** – With this option you can create a root Microsoft Enterprise CA in the AWS Cloud using Amazon EC2 and join it to your AWS Microsoft AD domain. This root CA can issue the certificate to your domain controllers. For more information about setting up a new root CA, see Install a Root Certification Authority on the Microsoft TechNet website.

For more information about how to join your EC2 instance to the domain, see Add an Instance to Your Directory (Simple AD and Microsoft AD) (p. 155).

Step 3: Create a Certificate Template

After your Enterprise CA has been set up, you can create a custom LDAPS certificate template in Active Directory. The certificate template must be created with Server Authentication and Autoenroll enabled. For more information, see Create a New Certificate Template on the Microsoft TechNet website.

To create a certificate template

1. Log in to your CA with Admin credentials
2. Launch **Server Manager.**, and then choose **Tools**, **Certification Authority**.
3. In the **Certificate Authority** window, expand your CA tree in the left pane. Right-click **Certificate Templates** and then choose **Manage**.
4. In the **Certificate Templates Console** window, right-click **Domain Controller**, and then choose **Duplicate Template**.
5. In the **Properties of New Template** window, switch to the **General** tab and change the **Template display name** to `ServerAuthentication`.
6. Switch to the **Security** tab and choose **Domain Controllers** in the **Group or user names** section. Select the **Autoenroll** check box in the **Permissions forDomain Controllers** section.
7. Switch to the **Extensions** tab, choose **Application Policies** in the **Extensions included in this template** section, and then choose **Edit**.
8. In the **Edit Application Policies Extension** window, choose **Client Authentication** and choose **Remove**. Click **OK** to create the `ServerAuthentication` certificate template, and then close the **Certificate Templates Console** window.
9. In the **Certificate Authority** window, right-click **Certificate Templates**, and choose **New**, **Certificate Template to Issue**.

10. In the **Enable Certificate Templates** window, choose `ServerAuthentication`, and then click **OK**.

Step 4: Add Security Group Rules

In the final step, you must open the Amazon EC2 console and add security group rules so that your domain controllers can connect to your Enterprise CA to request a certificate. To do this, you add inbound rules so that your Enterprise CA can accept incoming traffic from your domain controllers. Then you add outbound rules to allow traffic from your domain controllers to the Enterprise CA.

Once both rules have been configured, your domain controllers request a certificate from your Enterprise CA automatically and enable LDAPS for your directory. The LDAP service on your domain controllers is now ready to accept LDAPS connections.

To configure security group rules

1. Navigate to your Amazon EC2 console at https://console.aws.amazon.com/ec2 and sign in with Admin credentials.
2. In the left pane, choose **Security Groups** under **Network & Security**.
3. In the main pane, choose the AWS security group for your CA.
4. Choose the **Inbound** tab, and then choose **Edit**.
5. In the **Edit inbound rules** dialog box, do the following:
   - Choose **Add Rule**.
   - Choose **All traffic** for **Type** and **Custom** for **Source**.
   - Type your CA's AWS security group in the box next to **Source**.
   - Choose **Save**.
6. Now choose the AWS security group of your AWS Microsoft AD directory. Choose the **Outbound** tab and then choose **Edit**.
7. In the **Edit outbound rules** dialog box, do the following:
   - Choose **Add Rule**.
   - Choose **All traffic** for **Type** and **Custom** for **Destination**.
   - Type your CA's AWS security group in the box next to **Destination**.
   - Choose **Save**.

You can test the LDAPS connection to the AWS Microsoft AD directory using the LDP tool. The LDP tool comes with the Active Directory Administrative Tools. For more information, see Installing the Active Directory Administration Tools (p. 148).

> **Note**
> Before you test the LDAPS connection, you must wait up to 180 minutes for the subordinate CA to issue a certificate to your domain controllers.

For additional details about LDAPS and to see an example use case on how to set it up, see How to Enable LDAPS for Your AWS Microsoft AD Directory on the AWS Security Blog.

Multi-Factor Authentication

Multi-factor authentication (MFA) can be enabled for your Microsoft AD and AD Connector directories to help add an extra layer of protection on top of standard username and password authentication mechanisms. When MFA is enabled, users are required to enter an authentication code (the second factor) which is provided by your virtual or hardware MFA solution, in addition to entering their username and password (the first factor). These factors together provide additional security by preventing access to your Amazon Enterprise Applications, unless users supply a valid MFA code.

Supported Amazon Enterprise Applications

All Amazon Enterprise IT Applications including Amazon WorkSpaces, Amazon WorkDocs, Amazon WorkMail, Amazon QuickSight, and access to the AWS Management Console are supported when using Microsoft AD and AD Connector with MFA.

For information about how to configure basic user access to Amazon Enterprise Applications and the AWS Management Console using AWS Directory Service, see Manage Access to AWS Applications and Services (p. 144) and Manage Access to the AWS Management Console (p. 145).

Topics

Related AWS Security Blog Article

- How to Enable Multi-Factor Authentication for AWS Services by Using Microsoft AD and On-Premises Credentials

Enable Multi-Factor Authentication for Microsoft AD

To enable MFA for users of Amazon Enterprise Applications, a key requirement is an MFA solution that is a Remote Authentication Dial-In User Service (RADIUS) server or a plugin to a RADIUS server already implemented in your on-premises infrastructure. RADIUS is an industry-standard client/server protocol that provides authentication, authorization, and accounting management to enable users to connect network services. The RADIUS server connects to your on-premises AD to authenticate and authorize users. You can enable multi-factor authentication for your Microsoft AD directory by performing the following procedure.

For more information about how to configure your RADIUS server to work with AWS Directory Service and MFA, see Multi-factor Authentication Prerequisites (p. 57).

> **Note**
> Multi-factor authentication is not available for Simple AD.

To enable multi-factor authentication for Microsoft AD

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. Choose the directory ID link for your Microsoft AD directory.
3. Select the **Multi-Factor Authentication** tab.
4. Enter the following values, and then choose **Update Directory**.

   Enable Multi-Factor Authentication

      Check to enable multi-factor authentication.

   RADIUS server IP address(es)

      The IP addresses of your RADIUS server endpoints, or the IP address of your RADIUS server load balancer. You can enter multiple IP addresses by separating them with a comma (e.g., `192.0.0.0,192.0.0.12`).

      > **Note**
      > RADIUS MFA is applicable only to authenticate access to the AWS Management Console, or to Amazon Enterprise applications and services such as Amazon WorkSpaces, Amazon QuickSight, or Amazon Chime. AWS Directory Service does not

support RADIUS Challenge/Response authentication. Users must have their MFA code at the time they enter their username and password. Alternatively, you must use a solution that performs MFA out-of-band such as SMS text verification for the user.

Port

The port that your RADIUS server is using for communications. Your on-premises network must allow inbound traffic over the default RADIUS server port (1812) from the AWS Directory Service servers.

Shared secret code

The shared secret code that was specified when your RADIUS endpoints were created.

Confirm shared secret code

Confirm the shared secret code for your RADIUS endpoints.

Protocol

Select the protocol that was specified when your RADIUS endpoints were created.

Server timeout

The amount of time, in seconds, to wait for the RADIUS server to respond. This must be a value between 1 and 50.

Max retries

The number of times that communication with the RADIUS server is attempted. This must be a value between 0 and 10.

Multi-factor authentication is available when the **RADIUS Status** changes to **Enabled**.

Manage Fine-Grained Password Policies in Microsoft AD

AWS Microsoft AD enables you to define and assign different password and account lockout policies (also referred to as fine-grained password policies) for groups of users you manage in your AWS Microsoft AD domain.

For example, you can assign a less strict policy setting for employees that have access to low sensitivity information only. For senior managers who regularly access confidential information you can apply more strict settings.

AWS provides a set of fine-grained password policies in AWS Microsoft AD that you can configure and assign to your groups. To configure the policies, you can use standard Microsoft policy tools such as Active Directory Administrative Center (ADAC). To get started with the Microsoft policy tools, see Installing the Active Directory Administration Tools (p. 148).

Topics

Related AWS Security Blog Article

- How to Configure Even Stronger Password Policies to Help Meet Your Security Standards by Using AWS Directory Service for Microsoft AD

Supported Policy Settings

AWS Microsoft AD includes five fine-grained policies with a non-editable precedence value. The policies have a number of properties you can configure to enforce the strength of passwords, and account lock-out actions in the event of login failures. You can assign the policies to zero or more Active Directory groups. If an end-user is a member of multiple groups and receives more than one password policy, Active Directory enforces the policy with the lowest precedence value.

AWS Pre-Defined Password Policies

The following table lists the five policies included in your AWS Microsoft AD directory and their assigned precedence value. For more information, see Precedence (p. 113).

Policy name	Precedence
CustomerPSO-01	10
CustomerPSO-02	20
CustomerPSO-03	30
CustomerPSO-04	40
CustomerPSO-05	50

Password Policy Properties

You may edit the following properties in your password policies to conform to the compliance standards that meet your business needs.

- Policy name
- Password history
- Minimum password length
- Minimum password age
- Maximum password age
- Store password using reversible encryption
- Password must meet complexity requirements

You cannot modify the precedence values for these policies. For more details about how these settings affect password enforcement, see AD DS: Fine-Grained Password Policies on the *Microsoft TechNet* website. For general information about these policies, see Password Policy on the *Microsoft TechNet* website.

Account Lockout Policies

You may also modify the following properties of your password policies to specify if and how Active Directory should lockout an account after login failures:

- Number of failed logon attempts allowed
- Account lockout duration
- Reset failed logon attempts after some duration

For general information about these policies, see Account Lockout Policy on the *Microsoft TechNet* website.

Precedence

Policies with a lower precedence value have higher priority. You assign password policies to Active Directory security groups. While you should apply a single policy to a security group, a single user may receive more than one password policy. For example, suppose `jsmith` is a member of the HR group and also a member of the MANAGERS group. If you assign **CustomerPSO-05** (which has a precedence of 50) to the HR group, and **CustomerPSO-04** (which has a precedence of 40) to MANAGERS, **CustomerPSO-04** has the higher priority and Active Directory applies that policy to `jsmith`.

If you assign multiple policies to a user or group, Active Directory determines the resultant policy as follows:

1. A policy you assign directly to the user object applies.
2. If no policy is assigned directly to the user object, the policy with the lowest precedence value of all policies received by the user as a result of group membership applies.

For additional details, see AD DS: Fine-Grained Password Policies on the *Microsoft TechNet* website.

Delegate Who Can Manage Your Password Policies

You can delegate permissions to manage password policies to specific user accounts you created in your AWS Microsoft AD by adding the accounts to the **Fine-Grained PWD Policy Admins** security group. When an account becomes a member of this group, the account has permissions to edit and configure any of the password policies listed previously (p. 112).

To delegate who can manage password policies

1. Launch Active Directory Administrative Center (ADAC) from any managed EC2 instance that you joined to your Microsoft AD domain.
2. Switch to the **Tree View** and navigate to **CORP\Users**.
3. Find the **Fine Grained PWD Policy Admins** user group. Add any users or groups from your domain to this group.

Assign Password Policies to Your Users

User accounts that are a member of the **Fine-Grained PWD Policy Admins** security group can use the following procedure to assign policies to users and security groups.

To assign password policies to your users

1. Launch Active Directory Administrative Center (ADAC) from any managed EC2 instance that you joined to your Microsoft AD domain.
2. Switch to the **Tree View** and navigate to **System\Password Settings Container**.
3. Double click on the fine-grained policy you want to edit. Click **Add** to edit the policy properties, and add users or security groups to the policy. For more information about the default fine-grained policies provided with AWS Microsoft AD, see AWS Pre-Defined Password Policies (p. 112).

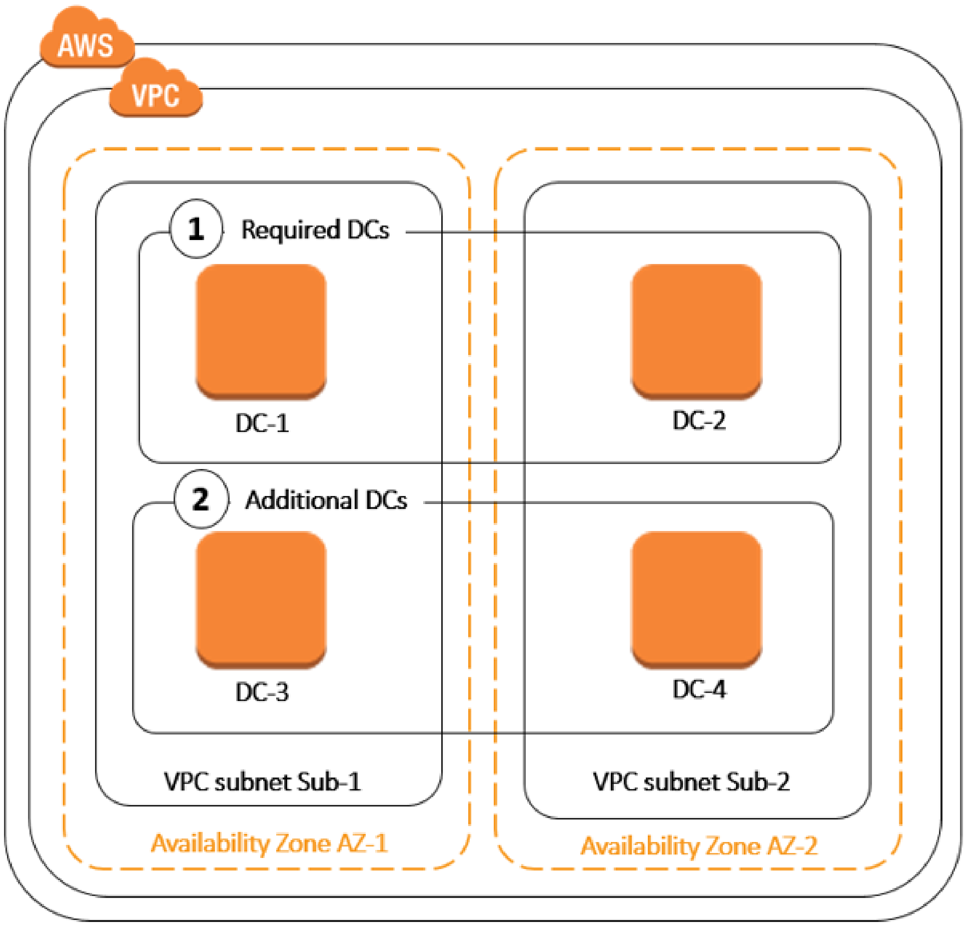
If you do not configure any of the five password policies in your AWS Microsoft AD directory, Active Directory uses the default domain group policy. For additional details on using **Password Settings Container**, see this Microsoft blog post.

Deploy Additional Domain Controllers

Deploying additional domain controllers increases the redundancy, which results in even greater resilience and higher availability. This also improves the performance of your directory by supporting a greater number of Active Directory requests. For example, you can now use AWS Microsoft AD to support multiple .NET applications that are deployed on large fleets of Amazon EC2 and Amazon RDS for SQL Server instances.

When you first create your directory, AWS Microsoft AD deploys two domain controllers across multiple Availability Zones, which is required for highly availability purposes. Later, you can easily deploy additional domain controllers via the AWS Directory Service console by just specifying the total number of domain controllers that you want. AWS Microsoft AD distributes the additional domain controllers to the Availability Zones and VPC subnets on which your directory is running.

For example, in the below illustration, DC-1 and DC-2 represent the two domain controllers that were originally created with your directory. The AWS Directory Service console refers to these default domain controllers as **Required**. Microsoft AD intentionally locates each of these domain controllers in separate Availability Zones during the directory creation process. Later, you might decide to add two more domain controllers to help distribute the authentication load over peak login times. Both DC-3 and DC-4 represent the new domain controllers, which the console now refers to as **Additional**. As before, Microsoft AD again automatically places the new domain controllers in different Availability Zones to ensure your domain's high availability.



This process eliminates the need for you to manually configure directory data replication, automated daily snapshots, or monitoring for the additional domain controllers. It's also easier for you to migrate and run mission critical Active Directory–integrated workloads in the AWS Cloud without having to deploy and maintain your own Active Directory infrastructure. You can also deploy or remove additional domain controllers for AWS Microsoft AD using the UpdateNumberOfDomainControllers API.

Add or Remove Additional Domain Controllers

Use the following procedure to deploy or remove additional domain controllers in your AWS Microsoft AD directory.

Note
If you have configured your AWS Microsoft AD to enable LDAPS, any additional domain controllers you add will also have LDAPS enabled automatically. For more information, see Secure LDAP Communications (p. 107).

To add or remove additional domain controllers

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Choose the directory ID link for your Microsoft AD directory.
3. On the **Details** page, choose the **Domain controllers** tab, and then choose **Modify**.
4. Specify the number of domain controllers to add or remove from your directory, and then choose **Apply**.
5. When AWS Microsoft AD completes the deployment process, all domain controllers show **Active** status, and both the assigned Availability Zone and VPC subnets appear. New domain controllers are equally distributed across the Availability Zones and subnets where your directory is already deployed.

Note
After deploying additional domain controllers, you can reduce the number of domain controllers to two, which is the minimum required for fault-tolerance and high availability purposes.

Related AWS Security Blog Article

- How to Increase the Redundancy and Performance of Your AWS Directory Service for Microsoft AD by Adding Domain Controllers

Manage Microsoft AD Compliance

You can use AWS Microsoft AD to support your Active Directory–aware applications, in the AWS Cloud, that are subject to the following compliance requirements. However, your applications will not adhere to compliance requirements if you use Simple AD or AD Connector.

Supported Compliance Standards

AWS Microsoft AD has undergone auditing for the following standards and is eligible for use as part of solutions for which you need to obtain compliance certification.

PCI Security Standards Council® PARTICIPATING ORGANIZATION™	AWS Microsoft AD has an Attestation of Compliance for Payment Card Industry (PCI) Data Security Standard (DSS) version 3.2 at Service Provider Level 1. Customers who use AWS products and services to store, process, or transmit cardholder data can use AWS Microsoft AD as they manage their own PCI DSS compliance certification. For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see PCI DSS Level 1. Importantly, you must configure fine-grained password policies in AWS Microsoft AD to be consistent with PCI DSS version 3.2 standards. For details on which policies must be enforced, see Enable PCI Compliance for Your AWS Microsoft AD Directory (p. 116).
HIPAA	AWS has expanded its Health Insurance Portability and Accountability Act (HIPAA) compliance program to include AWS Microsoft AD as a HIPAA Eligible Service. If you have an executed Business Associate Agreement (BAA) with AWS, you can use AWS Microsoft AD to help build your HIPAA-compliant applications. AWS offers a HIPAA-focused Whitepaper for customers who are interested in learning more about how they can leverage AWS for the processing and storage of health information. For more information, see HIPAA Compliance.

Shared Responsibility

Security, including HIPAA and PCI compliance, is a shared responsibility. It is important to understand that AWS Microsoft AD compliance status does not automatically apply to applications that you run in the AWS Cloud. You need to ensure that your use of AWS services complies with the standards.

Enable PCI Compliance for Your AWS Microsoft AD Directory

To enable PCI compliance for your AWS Microsoft AD directory, you must configure fine-grained password policies as specified in the PCI DSS Attestation of Compliance (AOC) and Responsibility Summary document provided by AWS Artifact.

For more information about using fine-grained password policies, see Manage Fine-Grained Password Policies in Microsoft AD (p. 111).

Security Logs

Security logs from AWS Microsoft AD domain controller instances are archived for a year. In the event of an audit to investigate a breach or other security event, you can request a copy of your security logs by contacting AWS Support.

AWS logs the following events for compliance.

Monitoring category	Policy setting	Audit state
Account Logon	Audit Credential Validation	Success, Failure
Account Management	Audit Computer Account Management	Success, Failure
	Audit Other Account Management Events	Success, Failure
	Audit Security Group Management	Success, Failure
	Audit User Account Management	Success, Failure
Detailed Tracking	Audit Process Creation	Success
DS Access	Audit Directory Service Access	Success, Failure
	Audit Directory Service Changes	Success, Failure
Logon/Logoff	Audit Account Lockout Success	Success, Failure
	Audit Logoff	Success
	Audit Logon	Success, Failure
	Audit Special Logon	Success
Object Access	Audit Removable Storage	Success, Failure
	Audit Central Access Policy Staging	Success, Failure
Policy Change	Audit Policy Change	Success, Failure
	Audit Authentication Policy Change	Success
	Audit Authorization Policy Change	Success, Failure
Privilege Use	Audit Sensitive Privilege Use	Success, Failure
System	Audit IPsec Driver	Success, Failure
	Audit Other System Events	Success, Failure
	Audit Security State Change	Success, Failure
	Audit Security System Extension	Success, Failure
	Audit System Integrity	Success, Failure

Active Directory Connector

AD Connector is a directory gateway with which you can redirect directory requests to your on-premises Microsoft Active Directory without caching any information in the cloud. AD Connector comes in two sizes, small and large. A small AD Connector is designed for smaller organizations of up to 500 users. A large AD Connector can support larger organizations of up to 5,000 users.

Once set up, AD Connector offers the following benefits:

- Your end users and IT administrators can use their existing corporate credentials to log on to AWS applications such as Amazon WorkSpaces, Amazon WorkDocs, or Amazon WorkMail.
- You can manage AWS resources like Amazon EC2 instances or Amazon S3 buckets through IAM role-based access to the AWS Management Console.
- You can consistently enforce existing security policies (such as password expiration, password history, and account lockouts) whether users or IT administrators are accessing resources in your on-premises infrastructure or in the AWS Cloud.
- You can use AD Connector to enable multi-factor authentication by integrating with your existing RADIUS-based MFA infrastructure to provide an additional layer of security when users access AWS applications.

Continue reading the topics in this section to learn how to connect to a directory and make the most of AD Connector features.

Topics

Connect to a Directory

With AD Connector you can connect AWS Directory Service to your existing enterprise directory. When connected to your on-premises directory, all of your directory data remains on your directory servers. AWS Directory Service does not replicate any of your directory data.

Topics

Best Practices for AD Connector

We recommend that you follow these best practices for creating your AD Connector:

- Each AD Connector that you create must use a different service account, even if they are connected to the same directory.
- If your on-premises network has Active Directory sites defined, you must make sure the subnets in the VPC where your AD Connector resides are defined in an Active Directory site, and that there are no conflicts between the subnets in your VPC and the subnets in your other sites. To discover domain controllers AD Connector uses the Active Directory site whose subnet IP address ranges are close to those in the VPC containing the AD Connector. If you have a site that has subnets with the same IP address ranges as those in your VPC, the AD Connector will discover the domain controllers in that site, which may not be physically close to your region.
- When using AD Connector, you must ensure that your on-premises directory is and remains compatible with AWS Directory Services. For more information on your responsibilities, please see our shared responsibility model.

AD Connector Prerequisites

To connect to your on-premises directory with AD Connector, you need the following:

VPC

Set up a VPC with the following:

- At least two subnets. Each of the subnets must be in a different Availability Zone.
- The VPC must be connected to your on-premises network through a virtual private network (VPN) connection or AWS Direct Connect.
- The VPC must have default hardware tenancy.

For more information, see the following topics in the *Amazon VPC User Guide*:

- What is Amazon VPC?
- Subnets in your VPC
- Adding a Hardware Virtual Private Gateway to Your VPC

For more information about AWS Direct Connect, see the AWS Direct Connect User Guide.

On-premises network

You'll need an on-premises network with an Active Directory domain. The functional level of this domain must be `Windows Server 2003` or higher. AD Connector also supports connecting to a domain hosted on an Amazon EC2 instance.

Note
AD Connector does not support Read-only domain controllers (RODC) when used in combination with the Amazon EC2 domain-join feature.

Credentials

You must have credentials for an account in the on-premises directory with the following privileges. For more information, see Delegating Connect Privileges (p. 120).

- Read users and groups
- Create computer objects
- Join computers to the domain

IP addresses

Get the IP addresses of two DNS servers or domain controllers in your on-premises directory.

AD Connector obtains the `_ldap._tcp.`*`<DnsDomainName>`* and `_kerberos._tcp.`*`<DnsDomainName>`* SRV records from these servers when connecting to your directory, so these servers must contain these SRV records. The AD Connector attempts to find a common domain controller that will provide both LDAP and Kerberos services, so these SRV records must include at least one common domain controller. For more information about SRV records, go to SRV Resource Records on Microsoft TechNet.

Ports for subnets

For AWS Directory Service to communicate with your on-premises directory, the firewall for your on-premises network must have the following ports open to the CIDRs for both subnets in the VPC.

- TCP/UDP 53 - DNS
- TCP/UDP 88 - Kerberos authentication
- TCP/UDP 389 - LDAP

These are the minimum ports that are needed to be able to connect to your directory. Your specific configuration may require additional ports be open.

Kerberos preauthentication

Your user accounts must have Kerberos preauthentication enabled. For more information about this setting, go to Preauthentication on Microsoft TechNet.

Encryption type

Your on-premises domain controller and user accounts must have RC4-HMAC encryption enabled.

Multi-factor Authentication Prerequisites

To support multi-factor authentication with your AD Connector directory, you need the following:

- A Remote Authentication Dial-In User Service (RADIUS) server in your on-premises network that has two client endpoints. The RADIUS client endpoints have the following requirements:
  - To create the endpoints, you need the IP addresses of the AWS Directory Service servers. These IP addresses can be obtained from the **Directory IP Address** field of your directory details.
  - Both RADIUS endpoints must use the same shared secret code.
- Your on-premises network must allow inbound traffic over the default RADIUS server port (1812) from the AWS Directory Service servers.
- The usernames between your RADIUS server and your on-premises directory must be identical.

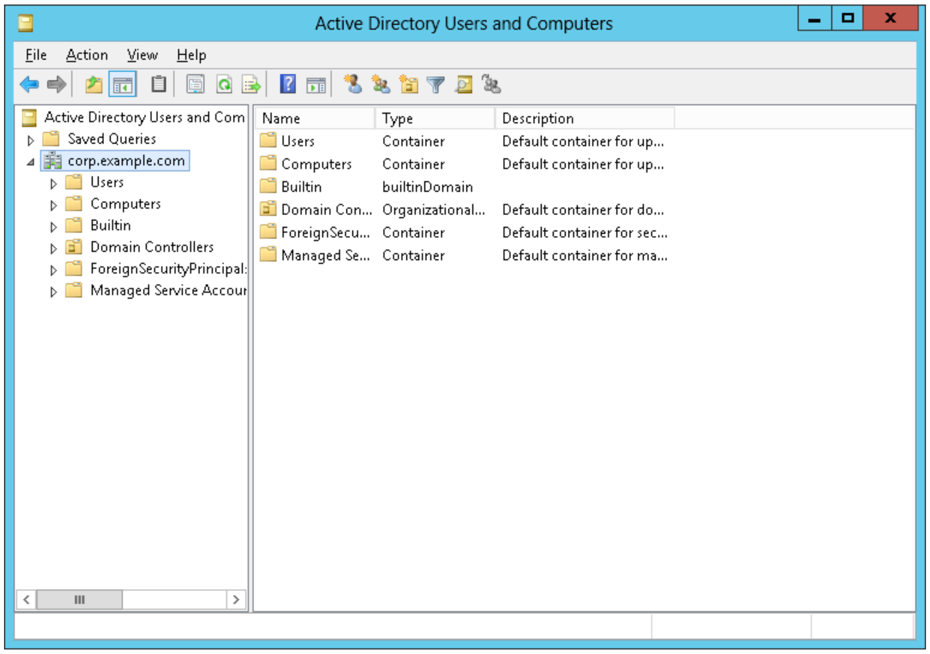
Delegating Connect Privileges

To connect to your on-premises directory, you must have the credentials for an account in the on-premises directory that has certain privileges. While members of the **Domain Admins** group have sufficient privileges to connect to the directory, as a best practice, you should use an account that only has the minimum privileges necessary to connect to the directory. The following procedure demonstrates how to create a new group called `Connectors`, and delegate the privileges to this group that are needed to connect AWS Directory Service to the directory.

This procedure must be performed on a machine that is joined to your directory and has the **Active Directory User and Computers** MMC snap-in installed. You must also be logged in as a domain administrator.

To delegate connect privileges

1. Open **Active Directory User and Computers** and select your domain root in the navigation tree.

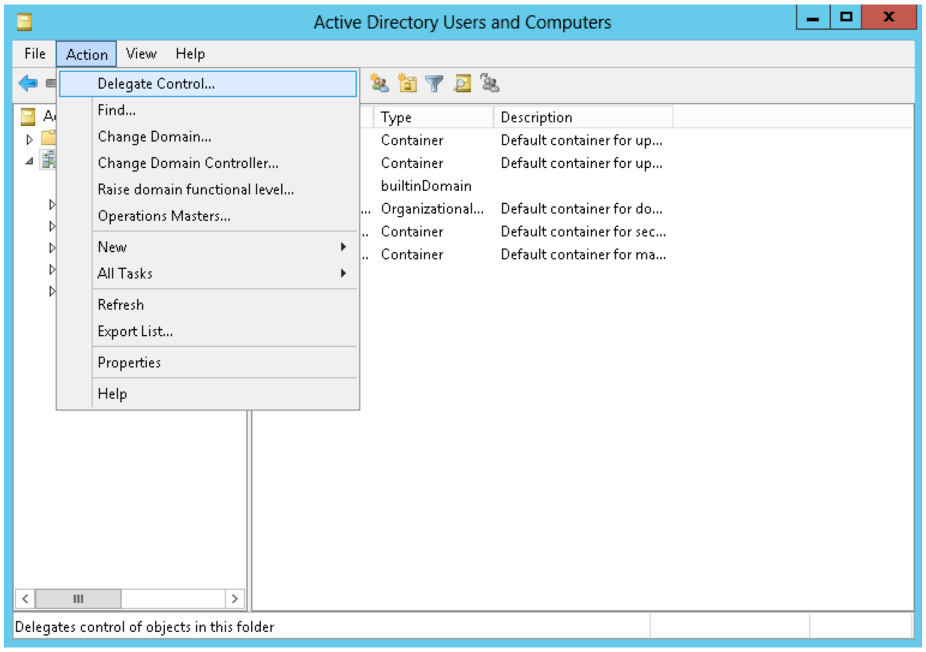


Name	Type	Description
Users	Container	Default container for up...
Computers	Container	Default container for up...
Builtin	builtinDomain	
Domain Con...	Organizational...	Default container for do...
ForeignSecu...	Container	Default container for sec...
Managed Se...	Container	Default container for ma...

2. In the list in the left-hand pane, right-click **Users**, select **New**, and then select **Group**.
3. In the **New Object - Group** dialog box, enter the following and click **OK**.

Field	Value/Selection
Group name	`Connectors`
Group scope	**Global**
Group type	**Security**

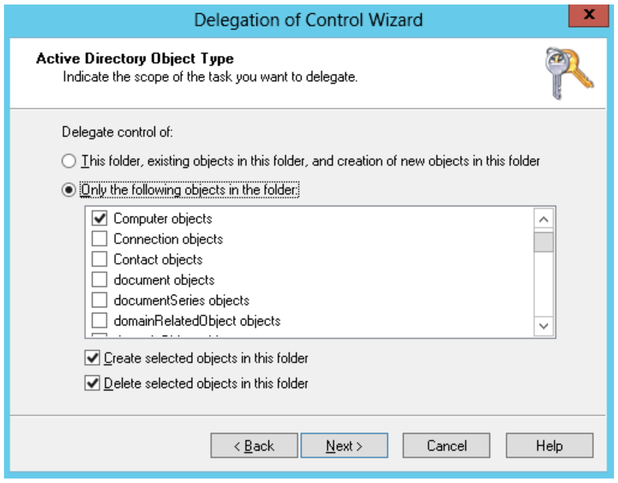
4. In the **Active Directory User and Computers** navigation tree, select your domain root. In the menu, select **Action**, and then **Delegate Control**.



Type	Description
Container	Default container for up...
Container	Default container for up...
builtinDomain	
Organizational...	Default container for do...
Container	Default container for sec...
Container	Default container for ma...

Delegates control of objects in this folder

5. On the **Delegation of Control Wizard** page, click **Next**, then click **Add**.
6. In the **Select Users, Computers, or Groups** dialog box, enter `Connectors` and click **OK**. If more than one object is found, select the `Connectors` group created above. Click **Next**.
7. On the **Tasks to Delegate** page, select **Create a custom task to delegate**, and then choose **Next**.
8. Select **Only the following objects in the folder**, and then select **Computer objects** and **User objects**.
9. Select **Create selected objects in this folder** and **Delete selected objects in this folder**. Then choose **Next**.

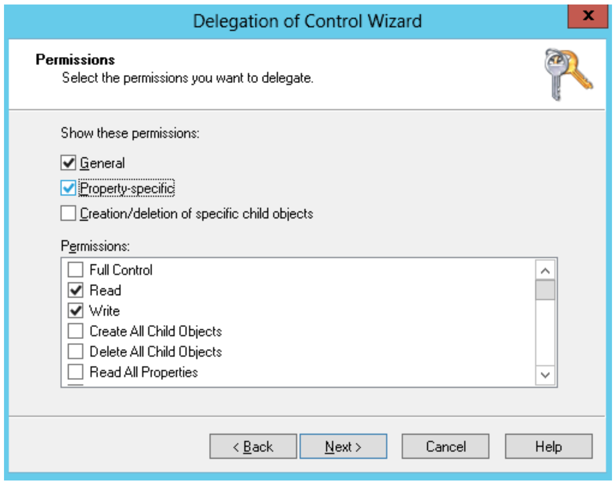


Active Directory Object Type
Indicate the scope of the task you want to delegate.

Delegate control of:

- ○ This folder, existing objects in this folder, and creation of new objects in this folder
- ● Only the following objects in the folder:
  - ☑ Computer objects
  - ☐ Connection objects
  - ☐ Contact objects
  - ☐ document objects
  - ☐ documentSeries objects
  - ☐ domainRelatedObject objects
- ☑ Create selected objects in this folder
- ☑ Delete selected objects in this folder

10. Select **Read** and **Write**, and then choose **Next**.


Delegation of Control Wizard

Permissions
Select the permissions you want to delegate.

Show these permissions:

- [x] General
- [x] Property-specific
- [ ] Creation/deletion of specific child objects

Permissions:

- [ ] Full Control
- [x] Read
- [x] Write
- [ ] Create All Child Objects
- [ ] Delete All Child Objects
- [ ] Read All Properties

11. Verify the information on the **Completing the Delegation of Control Wizard** page, and click **Finish**.
12. Create a user with a strong password and add that user to the `Connectors` group. The user has sufficient privileges to connect AWS Directory Service to the directory.

Connect Verification

For AD Connector to connect to your on-premises directory, the firewall for your on-premises network must have certain ports open to the CIDRs for both subnets in the VPC. To test if these conditions are met, perform the following steps:

To verify the connection

1. Launch a Windows instance in the VPC and connect to it over RDP. The instance must be a member of your on-premises domain. The remaining steps are performed on this VPC instance.
2. Download and unzip the DirectoryServicePortTest test application. The source code and Visual Studio project files are included so you can modify the test application if desired.

   Note
   This script is not supported on Windows Server 2003 or older operating systems.
3. From a Windows command prompt, run the **DirectoryServicePortTest** test application with the following options:

```
DirectoryServicePortTest.exe -d <domain_name> -ip <server_IP_address> -tcp
 "53,88,135,389,445,3268,5722,9389" -udp "53,88,123,138,389,445"
```

`<domain_name>`

   The fully qualified domain name. This is used to test the forest and domain functional levels. If you exclude the domain name, the functional levels won't be tested.

`<server_IP_address>`

   The IP address of a domain controller in your on-premises domain. The ports will be tested against this IP address. If you exclude the IP address, the ports won't be tested.

This test app determines if the necessary ports are open from the VPC to your domain, and also verifies the minimum forest and domain functional levels.

The output will be similar to the following:

```
Testing forest functional level.
Forest Functional Level = Windows2008R2Forest : PASSED

Testing domain functional level.
Domain Functional Level = Windows2008R2Domain : PASSED

Testing required TCP ports to <server_IP_address>:
Checking TCP port 53: PASSED
Checking TCP port 88: PASSED
Checking TCP port 389: PASSED

Testing required UDP ports to <server_IP_address>:
Checking UDP port 53: PASSED
Checking UDP port 88: PASSED
Checking UDP port 389: PASSED
```

The following is the source code for the **DirectoryServicePortTest** application.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;
using System.DirectoryServices.ActiveDirectory;
using System.Threading;
using System.DirectoryServices.AccountManagement;
using System.DirectoryServices;
using System.Security.Authentication;
using System.Security.AccessControl;
using System.Security.Principal;

namespace DirectoryServicePortTest
{
    class Program
    {
        private static List<int> _tcpPorts;
        private static List<int> _udpPorts;

        private static string _domain = "";
        private static IPAddress _ipAddr = null;

        static void Main(string[] args)
        {
            if (ParseArgs(args))
            {
                try
                {
                    if (_domain.Length > 0)
                    {
                        try
                        {
                            TestForestFunctionalLevel();

                            TestDomainFunctionalLevel();
                        }
                        catch (ActiveDirectoryObjectNotFoundException)
                        {
                            Console.WriteLine("The domain {0} could not be found.\n",
 _domain);
```

```
                    }
                }

                if (null != _ipAddr)
                {
                    if (_tcpPorts.Count > 0)
                    {
                        TestTcpPorts(_tcpPorts);
                    }

                    if (_udpPorts.Count > 0)
                    {
                        TestUdpPorts(_udpPorts);
                    }
                }
            }
            catch (AuthenticationException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        else
        {
            PrintUsage();
        }

        Console.Write("Press <enter> to continue.");
        Console.ReadLine();
    }

    static void PrintUsage()
    {
        string currentApp = 
Path.GetFileName(System.Reflection.Assembly.GetExecutingAssembly().Location);
        Console.WriteLine("Usage: {0} \n-d <domain> \n-ip \"<server IP address>\" 
\n[-tcp \"<tcp_port1>,<tcp_port2>,etc\"] \n[-udp \"<udp_port1>,<udp_port2>,etc\"]", 
currentApp);
    }

    static bool ParseArgs(string[] args)
    {
        bool fReturn = false;
        string ipAddress = "";

        try
        {
            _tcpPorts = new List<int>();
            _udpPorts = new List<int>();

            for (int i = 0; i < args.Length; i++)
            {
                string arg = args[i];

                if ("-tcp" == arg | "/tcp" == arg)
                {
                    i++;
                    string portList = args[i];
                    _tcpPorts = ParsePortList(portList);
                }

                if ("-udp" == arg | "/udp" == arg)
                {
                    i++;
                    string portList = args[i];
                    _udpPorts = ParsePortList(portList);
                }
```

```
                    if ("-d" == arg | "/d" == arg)
                    {
                        i++;
                        _domain = args[i];
                    }

                    if ("-ip" == arg | "/ip" == arg)
                    {
                        i++;
                        ipAddress = args[i];
                    }
                }
            }
            catch (ArgumentOutOfRangeException)
            {
                return false;
            }

            if (_domain.Length > 0 || ipAddress.Length > 0)
            {
                fReturn = true;
            }

            if (ipAddress.Length > 0)
            {
                _ipAddr = IPAddress.Parse(ipAddress);
            }

            return fReturn;
        }

        static List<int> ParsePortList(string portList)
        {
            List<int> ports = new List<int>();

            char[] separators = {',', ';', ':'};

            string[] portStrings = portList.Split(separators);
            foreach (string portString in portStrings)
            {
                try
                {
                    ports.Add(Convert.ToInt32(portString));
                }
                catch (FormatException)
                {
                }
            }

            return ports;
        }

        static void TestForestFunctionalLevel()
        {
            Console.WriteLine("Testing forest functional level.");

            DirectoryContext dirContext = new DirectoryContext(DirectoryContextType.Forest,
 _domain, null, null);
            Forest forestContext = Forest.GetForest(dirContext);

            Console.Write("Forest Functional Level = {0} : ", forestContext.ForestMode);

            if (forestContext.ForestMode >= ForestMode.Windows2003Forest)
            {
                Console.WriteLine("PASSED");
            }
            else
            {
                Console.WriteLine("FAILED");
            }

            Console.WriteLine();
        }

        static void TestDomainFunctionalLevel()
        {
            Console.WriteLine("Testing domain functional level.");

            DirectoryContext dirContext = new DirectoryContext(DirectoryContextType.Domain,
 _domain, null, null);
            Domain domainObject = Domain.GetDomain(dirContext);

            Console.Write("Domain Functional Level = {0} : ", domainObject.DomainMode);

            if (domainObject.DomainMode >= DomainMode.Windows2003Domain)
            {
                Console.WriteLine("PASSED");
            }
            else
            {
                Console.WriteLine("FAILED");
            }

            Console.WriteLine();
        }

        static List<int> TestTcpPorts(List<int> portList)
        {
            Console.WriteLine("Testing TCP ports to {0}:", _ipAddr.ToString());

            List<int> failedPorts = new List<int>();

            foreach (int port in portList)
            {
                Console.Write("Checking TCP port {0}: ", port);

                TcpClient tcpClient = new TcpClient();

                try
                {
                    tcpClient.Connect(_ipAddr, port);

                    tcpClient.Close();
                    Console.WriteLine("PASSED");
                }
                catch (SocketException)
                {
                    failedPorts.Add(port);
                    Console.WriteLine("FAILED");
                }
            }

            Console.WriteLine();

            return failedPorts;
        }

        static List<int> TestUdpPorts(List<int> portList)
        {
            Console.WriteLine("Testing UDP ports to {0}:", _ipAddr.ToString());
            List<int> failedPorts = new List<int>();

            foreach (int port in portList)
            {
                Console.Write("Checking UDP port {0}: ", port);

                UdpClient udpClient = new UdpClient();

                try
                {
                    udpClient.Connect(_ipAddr, port);
                    udpClient.Close();
                    Console.WriteLine("PASSED");
                }
                catch (SocketException)
                {
                    failedPorts.Add(port);
                    Console.WriteLine("FAILED");
                }
            }

            Console.WriteLine();

            return failedPorts;
        }
    }
}
```

How to Create an AD Connector

To connect to your on-premises directory with AD Connector, perform the following steps. Before starting this procedure, make sure you have completed the prerequisites identified in AD Connector Prerequisites (p. 119).

To connect with AD Connector

1. In the AWS Directory Service console navigation pane, select **Directories** and choose **Set up directory**.
2. Choose **AD Connector**.
3. Provide the following information:

   Connected directory DNS

   The fully qualified name of your on-premises directory, such as `corp.example.com`.

   Connected directory NetBIOS name

   The short name of your on-premises directory, such as `CORP`.

   Connector account username

   The user name of a user in the on-premises directory. For more information about this account, see the AD Connector Prerequisites (p. 119).

   Connector account password

   The password for the on-premises user account.

   Confirm password

   Retype the password for the on-premises user account.

DNS address

The IP address of at least one DNS server in your on-premises directory. These servers must be accessible from each subnet specified in the next section.

Description

An optional description for the directory.

Size

Select the size of the directory.

4. Provide the following information in the **VPC Details** section, and then choose **Next Step**.

   VPC

   The VPC for the directory.

   Subnets

   Select the subnets for the directory servers. The two subnets must be in different Availability Zones.
5. Review the directory information and make any necessary changes. When the information is correct, choose **Create AD Connector**.

It takes several minutes for your directory to be connected. When it has been successfully extended, the **Status** value changes to `Active`.

Update Directory Credentials for Your AD Connector

The AD Connector directory credentials represent the account that is used to access your on-premises directory. You can modify these account credentials by performing the following steps.

Note
If single sign-on is enabled for the directory, AWS Directory Service must transfer the service principal name (SPN) from the current service account to the new service account. If the current service account does not have permission to delete the SPN or the new service account does not have permission to add the SPN, you are prompted for the credentials of a directory account that does have permission to perform both actions. These credentials are only used to transfer the SPN and are not stored by the service.

To modify your account credentials for AD Connector

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. Click the directory ID link for your directory.
3. Select the **Connector Account** tab.
4. Enter the new user name and password, and click **Update Directory**.

Update the DNS Address for Your AD Connector

Use the following steps to update the DNS addresses that your AD Connector is pointing to.

> **Note**
> If you have an update in progress, you must wait until it is complete before submitting another update.

To update your DNS settings for AD Connector

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Choose the name of the directory to be updated.
3. Select the **DNS settings** tab.
4. Type the updated DNS IP addresses and choose **Update directory**.

Enable Multi-Factor Authentication for AD Connector

You can enable multi-factor authentication for your AD Connector directory by performing the following procedure. For more information about using multi-factor authentication with AWS Directory Service, see AD Connector Prerequisites (p. 119).

> **Note**
> Multi-factor authentication is not available for Simple AD.

To enable multi-factor authentication for AD Connector

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. Choose the directory ID link for your AD Connector directory.
3. Select the **Multi-Factor Authentication** tab.
4. Enter the following values, and then choose **Update Directory**.

   Enable Multi-Factor Authentication

   Check to enable multi-factor authentication.

   RADIUS server IP address(es)

   The IP addresses of your RADIUS server endpoints, or the IP address of your RADIUS server load balancer. You can enter multiple IP addresses by separating them with a comma (e.g., `192.0.0.0,192.0.0.12`).

   > **Note**
   > RADIUS MFA is applicable only to authenticate access to the AWS Management Console, or to Amazon Enterprise applications and services such as Amazon WorkSpaces, Amazon QuickSight, or Amazon Chime. AWS Directory Service does not support RADIUS Challenge/Response authentication. Users must have their MFA code at the time they enter their username and password. Alternatively, you must use a solution that performs MFA out-of-band such as SMS text verification for the user.

   Port

   The port that your RADIUS server is using for communications. Your on-premises network must allow inbound traffic over the default RADIUS server port (1812) from the AWS Directory Service servers.

   Shared secret code

   The shared secret code that was specified when your RADIUS endpoints were created.

Confirm shared secret code

Confirm the shared secret code for your RADIUS endpoints.

Protocol

Select the protocol that was specified when your RADIUS endpoints were created.

Server timeout

The amount of time, in seconds, to wait for the RADIUS server to respond. This must be a value between 1 and 50.

Max retries

The number of times that communication with the RADIUS server is attempted. This must be a value between 0 and 10.

Multi-factor authentication is available when the **RADIUS Status** changes to **Enabled**.

Simple Active Directory

Simple AD is a standalone managed directory that is powered by a Samba 4 Active Directory Compatible Server. It is available in two sizes.

- Small - Supports up to 500 users (approximately 2,000 objects including users, groups, and computers).
- Large - Supports up to 5,000 users (approximately 20,000 objects including users, groups, and computers).

Simple AD provides a subset of the features offered by Microsoft AD, including the ability to manage user accounts and group memberships, create and apply group policies, securely connect to Amazon EC2 instances, and provide Kerberos-based single sign-on (SSO). However, note that Simple AD does not support features such as trust relationships with other domains, Active Directory Administrative Center, PowerShell support, Active Directory recycle bin, group managed service accounts, and schema extensions for POSIX and Microsoft applications.

Simple AD offers many advantages:

- Simple AD makes it easier to manage Amazon EC2 instances running Linux and Windows and deploy Windows applications in the AWS Cloud.
- Many of the applications and tools that you use today that require Microsoft Active Directory support can be used with Simple AD.
- User accounts in Simple AD allow access to AWS applications such as Amazon WorkSpaces, Amazon WorkDocs, or Amazon WorkMail.
- You can manage AWS resources through IAM role–based access to the AWS Management Console.
- Daily automated snapshots enable point-in-time recovery.

Continue reading the topics in this section to learn how to create your own Simple AD.

Topics

Create a Simple AD Directory

Simple AD creates a fully managed, Samba-based directory in the AWS cloud. When you create a directory with Simple AD, AWS Directory Service creates two directory servers and DNS servers on your behalf. The directory servers are created in different subnets in a VPC; this redundancy helps ensures that your directory remains accessible even if a failure occurs.

Topics

Supported Applications

The following applications have been tested for use with AWS Directory Service Simple AD directories:

- Microsoft Internet Information Services (IIS) on the following platforms:
  - Windows Server 2003 R2
  - Windows Server 2008 R1
  - Windows Server 2008 R2
  - Windows Server 2012
  - Windows Server 2012 R2
- Microsoft SQL Server:
  - SQL Server 2005 R2 (Express, Web, and Standard editions)
  - SQL Server 2008 R2 (Express, Web, and Standard editions)
  - SQL Server 2012 (Express, Web, and Standard editions)
  - SQL Server 2014 (Express, Web, and Standard editions)
- Microsoft SharePoint:
  - SharePoint 2010 Foundation
  - SharePoint 2010 Enterprise
  - SharePoint 2013 Enterprise

Simple AD Prerequisites

To create a Simple AD directory, you need a VPC with the following:

- At least two subnets. Each of the subnets must be in a different Availability Zone.
- The following ports must be open between the two subnets that you deploy your directory into. This is necessary to allow the domain controllers that AWS Directory Service creates for you to communicate with each other.
  - TCP/UDP 53 - DNS
  - TCP/UDP 88 - Kerberos authentication
  - UDP 123 - NTP
  - TCP 135 - RPC
  - UDP 137-138 - Netlogon
  - TCP 139 - Netlogon
  - TCP/UDP 389 - LDAP
  - TCP/UDP 445 - SMB
  - TCP 873 - Rsync
  - TCP 3268 - Global Catalog
  - TCP/UDP 1024-65535 - Ephemeral ports for RPC
- The VPC must have default hardware tenancy.
- The following encryption types must be enabled in the directory:
  - RC4_HMAC_MD5
  - AES128_HMAC_SHA1
  - AES256_HMAC_SHA1
  - Future encryption types

How to Create a Simple AD Directory

To create a new directory, perform the following steps. Before starting this procedure, make sure you have completed the prerequisites identified in Simple AD Prerequisites (p. 133).

To create a Simple AD directory

1. In the AWS Directory Service console navigation pane, select **Directories** and choose **Set up directory**.
2. Choose **Simple AD**
3. Provide the following information:

   Organization name

      A unique organization name for your directory that will be used to register client devices.

      This field is only available if you are creating your directory as part of launching Amazon WorkSpaces.

   Directory DNS

      The fully qualified name for the directory, such as `corp.example.com`.

   NetBIOS name

      The short name for the directory, such as `CORP`.

   Administrator password

      The password for the directory administrator. The directory creation process creates an administrator account with the user name `Administrator` and this password.

      The directory administrator password is case-sensitive and must be between 8 and 64 characters in length, inclusive. It must also contain at least one character from three of the following four categories:

      - Lowercase letters (a-z)
      - Uppercase letters (A-Z)
      - Numbers (0-9)
      - Non-alphanumeric characters (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

   Confirm password

      Retype the administrator password.

   Description

      An optional description for the directory.

   Directory Size

      Select the size of the directory.

4. Provide the following information in the **VPC Details** section, and then choose **Next Step**.

   VPC

      The VPC for the directory.

   Subnets

      Select the subnets for the directory servers. The two subnets must be in different Availability Zones.

5. Review the directory information and make any necessary changes. When the information is correct, choose **Create Simple AD**.

It takes several minutes for the directory to be created. When it has been successfully created, the **Status** value changes to `Active`.

What Gets Created

When you create a directory with Simple AD, AWS Directory Service performs the following tasks on your behalf:

- Sets up a Samba-based directory within the VPC.
- Creates a directory administrator account with the user name `Administrator` and the specified password. You use this account to manage your directory.

  Important
  Be sure to save this password. AWS Directory Service does not store this password and it cannot be retrieved or reset.
- Creates a security group for the directory controllers.
- Creates an account with the name `AWSAdminD-xxxxxxxxx` that has domain admin privileges. This account is used by AWS Directory Service to perform automated operations for directory maintenance operations, such as taking directory snapshots and FSMO role transfers. The credentials for this account are securely stored by AWS Directory Service.

Tutorial: Create a Simple AD Directory

The following tutorial walks you through all of the steps necessary to set up an AWS Directory Service Simple AD directory. It is intended to get you started with AWS Directory Service quickly and easily, but is not intended to be used in a large-scale production environment.

Topics

Prerequisites

This tutorial assumes the following:

- You have an active AWS account. For more information about managing accounts and your directory, see Managing Your Directory (p. 139).
- Your account has not reached its limit of VPCs for the region in which you want to use AWS Directory Service. For more information about Amazon VPC, see What is Amazon VPC? and Subnets in Your VPC in the *Amazon VPC User Guide*.
- You do not have an existing VPC in the region with a CIDR of 10.0.0.0/16.

Step 1: Create and Configure Your VPC

The following sections demonstrate how to create and configure a VPC for use with AWS Directory Service.

Topics

Create a New VPC

This tutorial uses one of the VPC creation wizards to create the following:

- The VPC
- One of the subnets
- An Internet gateway

To create your VPC using the VPC wizard

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, click **VPC Dashboard**. If you do not already have any VPC resources, locate the **Your Virtual Private Cloud** area of the dashboard and click **Get started creating a VPC**. Otherwise, click **Start VPC Wizard**.
3. Select the second option, **VPC with a Single Public Subnet**, and then click **Select**.
4. Enter the following information into the wizard and click **Create VPC**.

   IP CIDR block

   `10.0.0.0/16`

   VPC name

   `ADS VPC`

   Public subnet

   `10.0.0.0/24`

   Availability Zone

   No Preference

   Subnet name

   `ADS Subnet 1`

   Enable DNS hostnames

   Leave default selection

   Hardware tenancy

   Default
5. It takes several minutes for the VPC to be created. After the VPC is created, proceed to the following section to add a second subnet.

Add a Second Subnet

AWS Directory Service requires two subnets in your VPC, and each subnet must be in a different Availability Zone. The VPC wizard only creates one subnet, so you must manually create the second subnet, and specify a different Availability Zone than the first subnet. Create the second subnet by performing the following steps.

To create a subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, select **Subnets**, select the subnet with the name `ADS Subnet 1`, and select the **Summary** tab at the bottom of the page. Make a note of the Availability Zone of this subnet.
3. Click **Create Subnet** and enter the following information in the **Create Subnet** dialog box and click **Yes, Create**.

   Name tag

   `ADS Subnet 2`

   VPC

   Select your VPC. This is the VPC with the name `ADS VPC`.

   Availability Zone

   Select any Availability Zone other than the one noted in step 2. The two subnets used by AWS Directory Service must reside in different Availability Zones.

   CIDR Block

   `10.0.1.0/24`

Step 2: Create Your Simple AD Directory

To create a new directory, perform the following steps. Before starting this procedure, make sure you have completed the prerequisites identified in Simple AD Prerequisites (p. 133).

To create a Simple AD directory

1. In the AWS Directory Service console navigation pane, select **Directories** and choose **Set up directory**.
2. Choose **Simple AD**
3. Provide the following information:

   Organization name

   A unique organization name for your directory that will be used to register client devices.

   This field is only available if you are creating your directory as part of launching Amazon WorkSpaces.

   Directory DNS

   The fully qualified name for the directory, such as `corp.example.com`.

   NetBIOS name

   The short name for the directory, such as `CORP`.

   Administrator password

   The password for the directory administrator. The directory creation process creates an administrator account with the user name `Administrator` and this password.

   The directory administrator password is case-sensitive and must be between 8 and 64 characters in length, inclusive. It must also contain at least one character from three of the following four categories:

   - Lowercase letters (a-z)

- Uppercase letters (A-Z)
- Numbers (0-9)
- Non-alphanumeric characters (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

Confirm password

Retype the administrator password.

Description

An optional description for the directory.

Directory Size

Select the size of the directory.

4. Provide the following information in the **VPC Details** section, and then choose **Next Step**.

   VPC

   The VPC for the directory.

   Subnets

   Select the subnets for the directory servers. The two subnets must be in different Availability Zones.
5. Review the directory information and make any necessary changes. When the information is correct, choose **Create Simple AD**.

It takes several minutes for the directory to be created. When it has been successfully created, the **Status** value changes to `Active`.

Managing Your Directory

You use the AWS Directory Service management console to perform certain directory-related actions, such as changing directory information or deleting an existing directory. After a directory is created, most administrative functions are performed with directory management tools, such as the Active Directory Administration Tools.

Note
Simple AD directories do not support Active Directory Web Services. Because of this, tools that rely on Active Directory Web Services, such as the Active Directory Administrative Center, do not work with your Simple AD directory.

Topics

View Directory Information

You can view basic information about a directory within the directories page, or more detailed information in the directory details page.

Basic Information

To view basic information about a directory, perform the following steps:

To view basic directory information

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. Click the arrow button next to your directory. Basic information about the directory is displayed below the directory entry in the list.

For more information about the **Status** field, see Directory Status (p. 187).

Detailed Information

To view more detailed information about a directory, perform the following steps:

To view detailed directory information

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. Click the directory ID link for your directory. Information about the directory is displayed in the **Directory Details** section.

For more information about the **Status** field, see Directory Status (p. 187).

Get Notified of Directory Status Updates Using Amazon SNS

Using Amazon Simple Notification Service (Amazon SNS), you can receive email or text (SMS) messages when the status of your directory changes. You get notified if your directory goes from an Active status to an Impaired or Inoperable status. You also receive a notification when the directory returns to an Active status.

How It Works

Amazon SNS uses “topics” to collect and distribute messages. Each topic has one or more subscribers who receive the messages that have been published to that topic. Using the steps below you can add AWS Directory Service as publisher to an Amazon SNS topic. When AWS Directory Service detects a change in your directory’s status, it publishes a message to that topic, which is then sent to the topic's subscribers.

You can associate multiple directories as publishers to a single topic. You can also add directory status messages to topics that you’ve previously created in Amazon SNS. You have detailed control over who can publish to and subscribe to a topic. For complete information about Amazon SNS, see What is Amazon SNS?.

To enable SNS messaging for your directory

1. Sign in to the AWS Management Console and open the AWS Directory Service console at https://console.aws.amazon.com/directoryservice/.
2. On the **Directories** page, choose your directory ID.
3. Choose the **Monitoring** tab and then **Create Notification**.
4. Choose **Create a new notification**. Alternatively, if you already have an existing SNS topic, you can choose **Associate with existing SNS topic** to send status messages from this directory to that topic.

   Note
   If you choose **Create a new notification** but then use the same topic name for an SNS topic that already exists, Amazon SNS does not create a new topic, but just adds the new subscription information to the existing topic.
   If you choose **Associate with existing SNS topic**, you will only be able to choose an SNS topic that is in the same region as the directory.
5. Choose the **Recipient type** and enter the **Recipient** contact information. If you enter a phone number for SMS, use numbers only. Do not include dashes, spaces, or parentheses.
6. (Optional) Choose **Advanced options** and type a name for your topic and an SNS display name. The display name is a short name up to 10 characters that is included in all SMS messages from this topic. When using the SMS option, the display name is required.

   Note
   If you are logged in using an IAM user or role that has only the DirectoryServiceFullAccess managed policy, your topic name must start with “DirectoryMonitoring”. If you’d like to further customize your topic name you’ll need additional privileges for SNS.
7. Choose **Add**.

If you want to designate additional SNS subscribers, such as an additional email address, Amazon SQS queues or AWS Lambda, you can do this from the Amazon SNS console at https://console.aws.amazon.com/sns/v2/home.

To remove directory status messages from a topic

1. Sign in to the AWS Management Console and open the AWS Directory Service console at https://console.aws.amazon.com/directoryservice/.
2. On the **Directories** page, choose your directory ID.
3. Choose the **Monitoring** tab and then choose an **SNS topic name**.
4. Choose **Remove**.

This removes your directory as a publisher to the selected SNS topic. If you want to delete the entire topic, you can do this from the Amazon SNS console at https://console.aws.amazon.com/sns/v2/home.

> **Note**
> Before deleting an Amazon SNS topic using the SNS console, you should ensure that a directory is not sending status messages to that topic.
> If you delete an Amazon SNS topic using the SNS console, this change will not immediately be reflected within the Directory Services console. You would only be notified the next time a directory publishes a notification to the deleted topic, in which case you would see an updated status on the directory's **Monitoring** tab indicating the topic could not be found.
> Therefore, to avoid missing important directory status messages, before deleting any topic that receives messages from AWS Directory Service, associate your directory with a different Amazon SNS topic.

Delete Your Directory

When a Simple AD or AWS Directory Service for Microsoft Active Directory directory is deleted, all of the directory data and snapshots are deleted and cannot be recovered. After the directory is deleted, all instances that are joined to the directory remain intact. You cannot, however, use your directory credentials to log in to these instances. You need to log in to these instances with a user account that is local to the instance.

When an AD Connector directory is deleted, your on-premises directory remains intact. All instances that are joined to the directory also remain intact and remain joined to your on-premises directory. You can still use your directory credentials to log in to these instances.

To delete a directory

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. Ensure that no AWS applications are enabled for the directory.
    1. Click the directory ID link for your directory.
    2. Select the **Apps & Services** tab. In the Apps & Services section, you see which AWS applications are enabled for your directory.
        - To disable Amazon WorkSpaces, you must deregister the service from the directory in the Amazon WorkSpaces console. For more information, see Deregistering From a Directory in the *Amazon WorkSpaces Administration Guide*.
        - To disable Amazon WorkSpaces Application Manager, you must remove all application assignments in the Amazon WAM console. For more information, see Removing All Application Assignments in the *Amazon WAM Administration Guide*.
        - To disable Amazon WorkDocs, you must delete the Amazon WorkDocs site in the Amazon WorkDocs console. For more information, see Delete a Site in the *Amazon WorkDocs Administration Guide*.
        - To disable Amazon WorkMail, you must remove the Amazon WorkMail organization in the Amazon WorkMail console. For more information, see Remove an Organization in the *Amazon WorkMail Administrator Guide*.

- To disable AWS Management Console access, see Disable AWS Management Console Access (p. 146).
- To disable Amazon Relational Database Service, you must remove the Amazon RDS instance from the domain. For more information, see Managing a DB Instance in a Domain in the *Amazon Relational Database Service User Guide*.
- To disable Amazon QuickSight, you must unsubscribe from Amazon QuickSight. For more information, see Closing Your Amazon QuickSight Account in the *Amazon QuickSight User Guide*.
- To disable Amazon Connect, you must delete the Amazon Connect Instance. For more information, see Deleting an Amazon Connect Instance in the *Amazon Connect Administration Guide*.

3. In the navigation pane, choose **Directories**.
4. Select only the directory to be deleted and click **Delete**. It takes several minutes for the directory to be deleted. When the directory has been deleted, it is removed from your directory list.

Note
Before deleting a directory that is associated with an Amazon RDS database, you must first remove that database from the directory.

Snapshots (Simple AD and Microsoft AD)

AWS Directory Service provides the ability to take manual snapshots of data for a Simple AD or AWS Directory Service for Microsoft Active Directory directory. These snapshots can be used to perform a point-in-time restore for your directory.

Note
You cannot take snapshots of AD Connector directories.

Topics

Creating a Snapshot of Your Directory

A snapshot can be used to restore your directory to what it was at the point in time that the snapshot was taken. To create a manual snapshot of your directory, perform the following steps.

Note
You are limited to 5 manual snapshots for each directory. If you have already reached this limit, you must delete one of your existing manual snapshots before you can create another.

To create a manual snapshot

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. In the **Directory ID** column, choose the link for your directory.
3. Choose the **Snapshots** tab, and then choose **Create Snapshot**.
4. In the **Create directory snapshot** dialog box, provide a description of the snapshot, if desired. When ready, choose **Create Snapshot**.

Depending on the size of your directory, it may take several minutes to create the snapshot. When the snapshot is ready, the **Status** value changes to `Completed`.

Restoring Your Directory from a Snapshot

Restoring a directory from a snapshot is equivalent to moving the directory back in time.

Warning
Before you restore a directory from a snapshot, it is important you understand that all of the DCs and DNS servers associated with the directory will be offline until the restore operation has been completed.

To restore your directory from a snapshot, perform the following steps.

To restore a directory from a snapshot

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. In the **Directory ID** column, choose the link for your directory.
3. Choose the **Snapshots** tab, and then select the snapshot to restore from.
4. Choose **Restore**, review the information in the dialog box, and choose **Restore**.

For a Simple AD directory, it may take several minutes for the directory to be restored. For a Microsoft AD directory, it can take from two to three hours. When it has been successfully restored, the **Status** value of the directory changes to `Active`. Any changes made to the directory after the snapshot date are overwritten.

Deleting a Snapshot

To delete a snapshot

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. In the **Directory ID** column, choose the link for your directory.
3. Choose the **Snapshots** tab, and then select the snapshot to delete.
4. Choose **Delete**, verify that you want to delete the snapshot, and choose **Delete**.

Use Your Directory

Topics

Creating an Access URL

An Access URL is used with AWS applications and services, such as Amazon WorkSpaces, to reach a login page that is associated with your directory. The URL must be unique globally. You can create an access URL for your directory by performing the following steps.

> **Warning**
> After an access URL is created, it cannot be used by others. If you delete your directory, the access URL is also deleted and can then be used by any other account.

To create an access URL

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. Choose the directory ID link for your directory.
3. In the **Access URL** section, if an access URL has not been assigned to the directory, the **Create Access URL** button is displayed. Enter a directory alias and choose **Create Access URL**. If an **Entity Already Exists** error is returned, the specified directory alias has already been allocated. Choose another alias and repeat this procedure.

   Your directory URL is changed to *<alias>*.awsapps.com.

Manage Access to AWS Applications and Services

AWS Directory Service can give other AWS applications and services, such as Amazon WorkSpaces, access to your directory users. The following AWS applications and services can be enabled or disabled to work with AWS Directory Service:

Amazon WorkSpaces

You can create a Simple AD, Microsoft AD, or AD Connector directly from Amazon WorkSpaces. Simply launch **Advanced Setup** when creating your Workspace.

For more information, see the Amazon WorkSpaces Administration Guide.

Amazon WorkSpaces Application Manager

For more information, see the Amazon WAM Administration Guide.

Amazon WorkDocs

For more information, see the Amazon WorkDocs Administration Guide.

Amazon WorkMail

For more information, see the Amazon WorkMail Administrator Guide.

Amazon Relational Database Service

For more information, see the Amazon Relational Database Service User Guide.

AWS Management Console

For more information, see Manage Access to the AWS Management Console (p. 145).

Once enabled, you manage access to your directories in the console of the application or service that you want to give access to your directory. To find the AWS applications and services links described above in the AWS Directory Service console, perform the following steps.

To display the applications and services for a directory

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Choose the directory ID link for your directory.
3. Choose the **Apps & Services** tab.

Manage Access to the AWS Management Console

AWS Directory Service allows you to grant members of your directory access to the AWS Management Console. By default, your directory members do not have access to any AWS resources. You assign IAM roles to your directory members to give them access to the various AWS services and resources. The IAM role defines the services, resources, and level of access that your directory members have.

Before you can grant console access to your directory members, your directory must have an access URL. For more information about how to view directory details and get your access URL, see View Directory Information (p. 139). For more information about how to create an access URL, see Creating an Access URL (p. 144).

For more information about how to create and assign IAM roles to your directory members, see Grant Users and Groups Access to AWS Resources (p. 152).

Topics

Related AWS Security Blog Article

- How to Access the AWS Management Console Using Microsoft AD and Your On-Premises Credentials

Enable AWS Management Console Access

By default, console access is not enabled for any directory. To enable console access for your directory users and groups, perform the following steps:

To enable console access

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Choose the directory ID for your directory.
3. In the **Directory Details** page, choose the **Apps & services** tab.
4. Under **AWS apps & services**, choose **AWS Management Console**.
5. In the **Enable AWS Management Console** dialog box, choose **Enable Access**. Console access is now enabled for your directory.
6. Before users can sign-in to the console with your access URL, you must add a **New Role** and add only those users who you want to have access. For more information about assigning users to IAM roles, see Creating a New Role (p. 153) or Assigning Users or Groups to an Existing Role (p. 153).

   After the IAM roles have been assigned, users can then access the console using your access URL. For example, if your directory's access URL is example-corp.awsapps.com, the URL to access the console is https://example-corp.awsapps.com/console/.

   Note
   Access for users in nested groups within your directory are not supported. Members of the parent group have console access, but members of child groups do not.

Disable AWS Management Console Access

To disable console access for your directory users and groups, perform the following steps:

To disable console access

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Choose the directory ID for your directory.
3. In the **Directory Details** page, choose the **Apps & services** tab.
4. Under **AWS apps & services**, choose **AWS Management Console**.
5. If any IAM roles have been assigned to users or groups in the directory, the **Disable Access** button in the **Manage access to AWS Resources** dialog box is unavailable. In this case, you must choose **Continue** and remove all IAM role assignments for the directory before proceeding, including assignments for users or groups in your directory that have been deleted, which will show as **Deleted User** or **Deleted Group**.

   After all IAM role assignments have been removed, repeat the steps above. When the **Manage access to AWS Resources** dialog box is displayed, choose **Disable Access**.

Set Login Session Length

By default, users have 1 hour to use their session after successfully signing in to the console before they are logged out. After that, users must sign in again to start the next 1 hour session before being logged off again. You can use the following procedure to change the length of time to up to 12 hours per session.

To set login session length

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Choose the directory ID for your directory.
3. In the **Directory Details** page, choose the **Apps & services** tab.
4. Under **AWS apps & services**, choose **AWS Management Console**.

5. In the **Manage Access to AWS Resource** dialog box, choose **Continue**.
6. In the **Assign users and groups to IAM roles** page, under **Set login session length**, edit the numbered value, and then choose **Save**.

Add Users and Groups (Simple AD and Microsoft AD)

Users represent individual people or entities that have access to your directory. Groups are very useful for giving or denying privileges to groups of users, rather than having to apply those privileges to each individual user. If a user moves to a different organization, you move that user to a different group and they automatically receive the privileges needed for the new organization.

To create users and groups in an AWS Directory Service directory, you must be connected to a EC2 instance that has been joined to your AWS Directory Service directory, and be logged in as a user that has privileges to create users and groups. You'll also need to install the Active Directory Tools on your EC2 instance so you can add your users and groups with the Active Directory Users and Computers snap-in. For more information about how to set up an EC2 instance and install the necessary tools, see Step 3: Deploy an EC2 Instance to Manage Microsoft AD (p. 98).

> **Note**
> Your user accounts must have Kerberos preauthentication enabled. This is the default setting for new user accounts, but it should not be modified. For more information about this setting, go to Preauthentication on Microsoft TechNet.

The following examples demonstrate how to create a user, create a group, and add the user to the group.

> **Note**
> When using Simple AD, if you create a user account on a Linux instance with the option "Force user to change password at first login," that user will not be able to initially change their password using **kpasswd**. In order to change the password the first time, a domain administrator must update the user password using the Active Directory Management Tools.

To create a user

1. Open the Active Directory Users and Computers tool. There is a shortcut to this tool in the **Administrative Tools** folder.

   > **Tip**
   > You can run the following from a command prompt on the instance to open the Active Directory Users and Computers tool box directly.

   ```
   %SystemRoot%\system32\dsa.msc
   ```
2. In the directory tree, open your directory and select the **Users** folder.
3. On the **Action** menu, click **New**, and then click **User** to open the new user wizard.
4. In the first page of the new user wizard, enter the following values and click **Next**.

   First name

   `Mary`

   Last name

   `Major`

   User logon name

   `marym`

5. In the second page of the new user wizard, enter a temporary password for **Password** and **Confirm Password**. Make sure the **User must change password at next logon** option is selected. None of the other options should be selected. Click **Next**.
6. In the third page of the new user wizard, verify that the new user information is correct and click **Finish**. The new user will appear in the **Users** folder.

To create a group

1. Open the Active Directory Users and Computers tool. There is a shortcut to this tool in the **Administrative Tools** folder.

   Tip
   You can run the following from a command prompt on the instance to open the Active Directory Users and Computers tool box directly.

   ```
   %SystemRoot%\system32\dsa.msc
   ```
2. In the directory tree, open your directory and select the **Users** folder.
3. On the **Action** menu, click **New**, and then click **Group** to open the new group wizard.
4. Enter **`Division Managers`** for the **Group name**, select **Global** for the **Group scope**, and select **Security** for the **Group type**. Click **OK**. The new group, **Division Managers**, appears in the **Users** folder.

To add a user to a group

1. Open the Active Directory Users and Computers tool. There is a shortcut to this tool in the **Administrative Tools** folder.

   Tip
   You can run the following from a command prompt on the instance to open the Active Directory Users and Computers tool box directly.

   ```
   %SystemRoot%\system32\dsa.msc
   ```
2. In the directory tree, open your directory, select the **Users** folder, and select the **Division Managers** group.
3. On the **Action** menu, click **Properties** to open the properties dialog box for the **Division Managers** group.
4. Select the **Members** tab and click **Add...**.
5. For **Enter the object names to select**, enter `marym` and click **OK**. **Mary Major** is displayed in the **Members** list. Click **OK** again to update the group membership.
6. Verify that Mary Major is now a member of the **Division Managers** group by selecting **Mary Major** in the **Users** folder, click **Properties** in the **Action** menu to open the properties dialog box for Mary Major. Select the **Member Of** tab. **Division Managers** is in the list of groups that Mary Major belongs to.

Installing the Active Directory Administration Tools

To manage your directory from an EC2 Windows instance, you need to install the Active Directory Domain Services and Active Directory Lightweight Directory Services Tools on the instance.

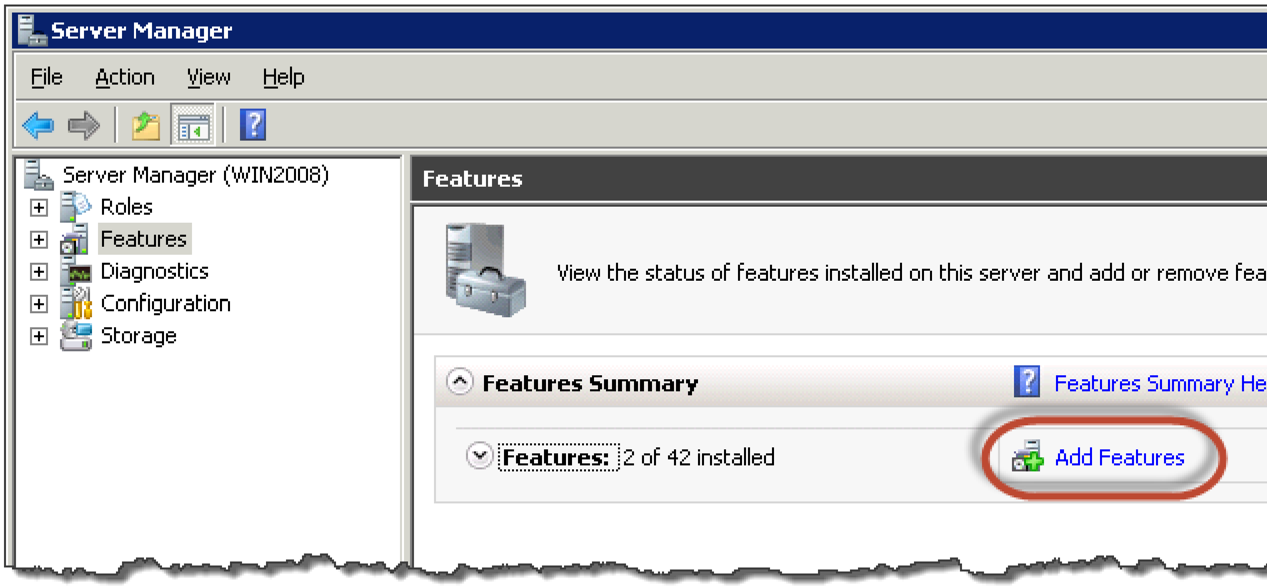
Topics

- Install the Active Directory Administration Tools on Windows Server 2012 (p. 150)

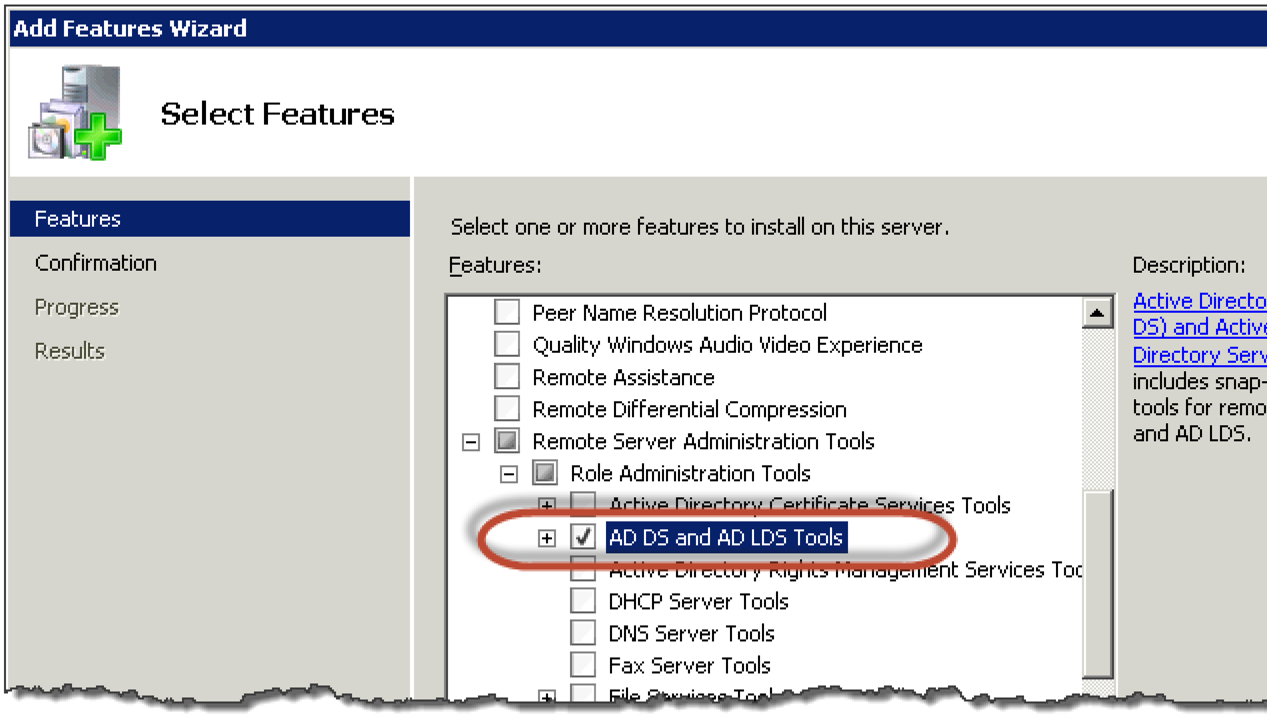
Install the Active Directory Administration Tools on Windows Server 2008

To install the Active Directory administration tools on Windows Server 2008

1. Open Server Manager by choosing **Start**, **Administrative Tools**, **Server Manager**.
2. In the **Server Manager** tree pane, select **Features**, and choose **Add Features**,



3. In the **Add Features Wizard**, open **Remote Server Administration Tools**, **Role Administration Tools**, select **AD DS and AD LDS Tools**, scroll down and select **DNS**, then choose **Next**.



Select one or more features to install on this server.

Features:

- [ ] Peer Name Resolution Protocol
- [ ] Quality Windows Audio Video Experience
- [ ] Remote Assistance
- [ ] Remote Differential Compression
- ▨ Remote Server Administration Tools
  - ▨ Role Administration Tools
    - [ ] Active Directory Certificate Services Tools
    - [✓] AD DS and AD LDS Tools
    - [ ] Active Directory Rights Management Services Toc
    - [ ] DHCP Server Tools
    - [ ] DNS Server Tools
    - [ ] Fax Server Tools
    - [ ] File Services Tool

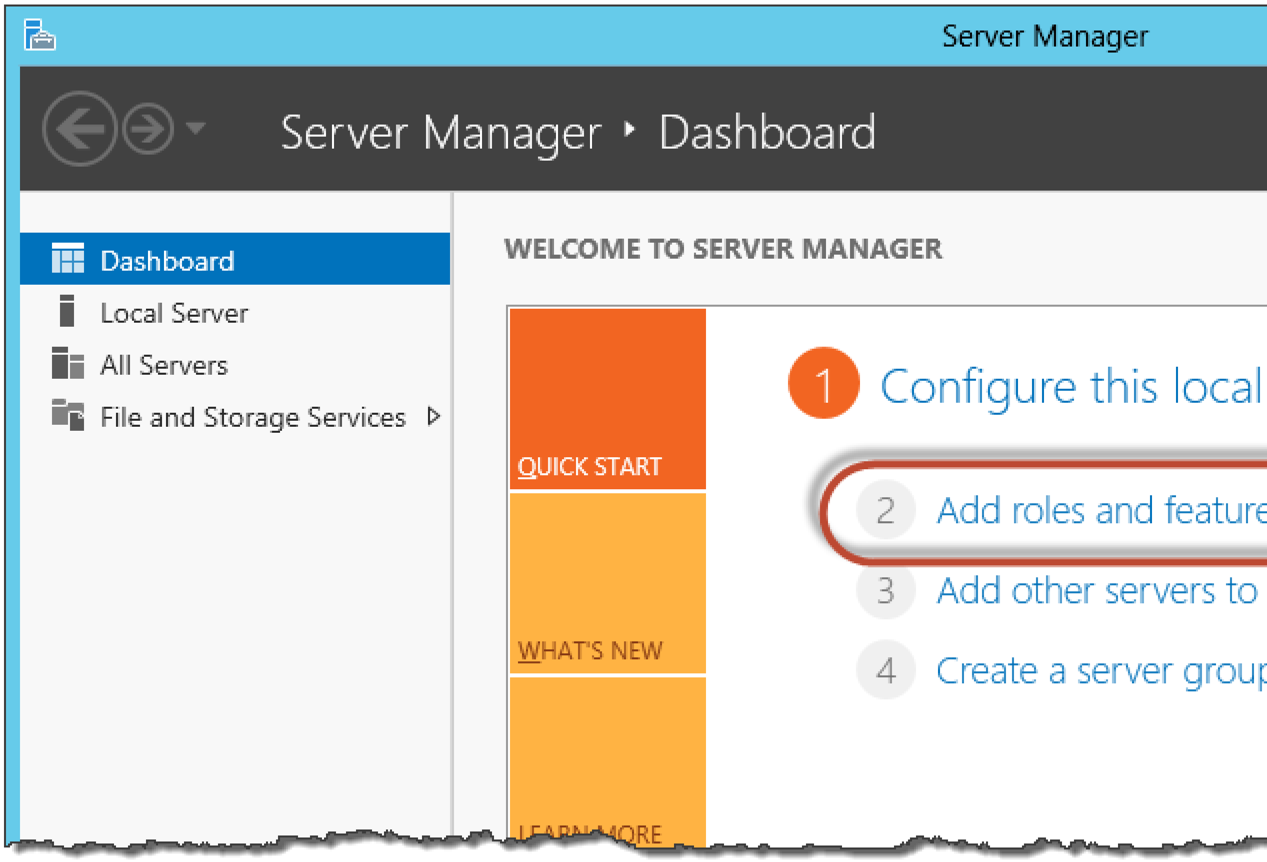
Active Directo
DS) and Active
Directory Serv
includes snap-
tools for remo
and AD LDS.

4. Review the information and choose **Install**. The feature installation requires that the instance be restarted. When the instance has restarted, the Active Directory Domain Services and Active Directory Lightweight Directory Services Tools are available on the **Start** menu, under **All Programs** > **Administrative Tools**.

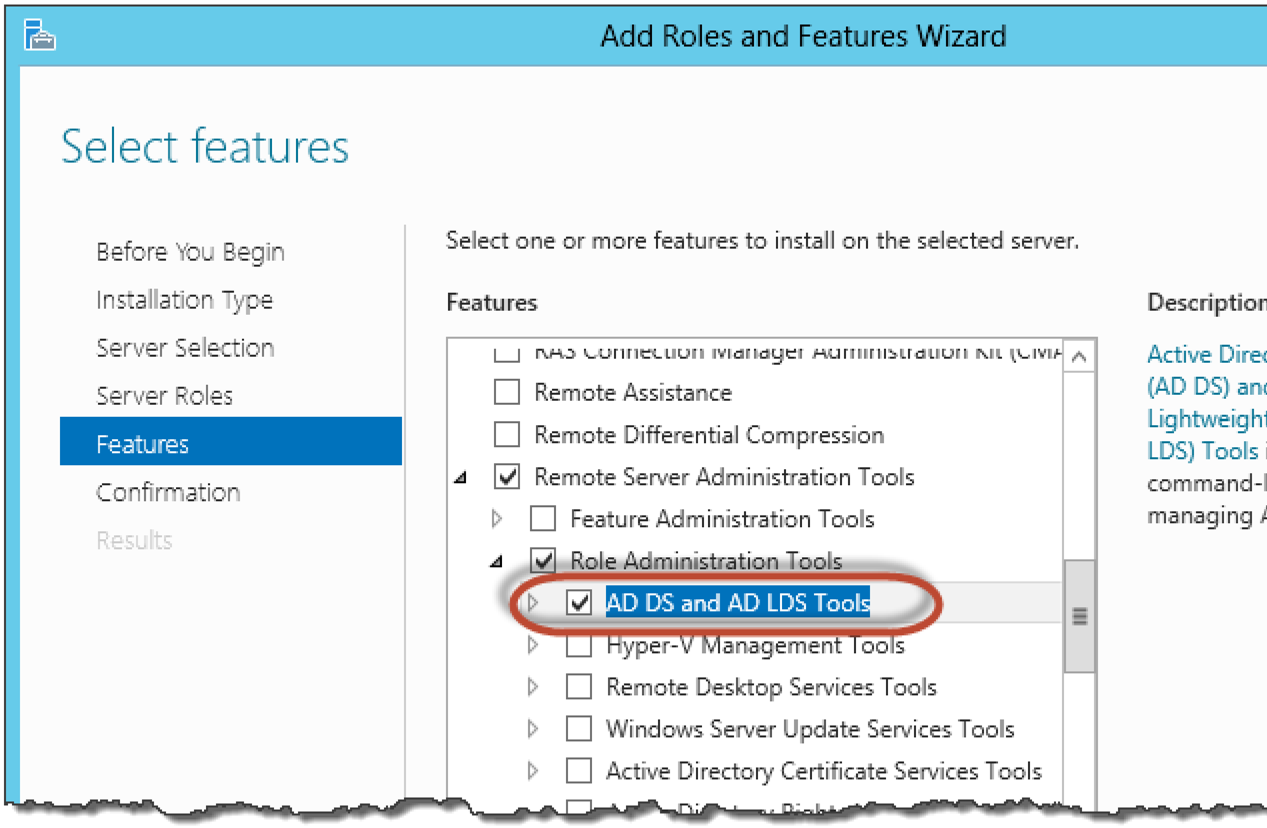
Install the Active Directory Administration Tools on Windows Server 2012

To install the Active Directory administration tools on Windows Server 2012

1. Open Server Manager by from the Start screen by choosing **Server Manager**.
2. In the **Server Manager Dashboard**, choose **Add roles and features**,



3. In the **Add Roles and Features Wizard** choose **Installation Type**, select **Role-based or feature-based installation**, and choose **Next**.
4. Under **Server Selection**, make sure the local server is selected, and choose **Features**.
5. In the **Features** tree, open **Remote Server Administration Tools**, **Role Administration Tools**, select **AD DS and AD LDS Tools**, scroll down and select **DNS**, then choose **Next**.



Select one or more features to install on the selected server.

Features

- [ ] RAS Connection Manager Administration Kit (CMA
- [ ] Remote Assistance
- [ ] Remote Differential Compression
- [✓] Remote Server Administration Tools
  - [ ] Feature Administration Tools
  - [✓] Role Administration Tools
    - [✓] AD DS and AD LDS Tools
    - [ ] Hyper-V Management Tools
    - [ ] Remote Desktop Services Tools
    - [ ] Windows Server Update Services Tools
    - [ ] Active Directory Certificate Services Tools

Description

Active Direc
(AD DS) and
Lightweight
LDS) Tools i
command-l
managing A

6. Review the information and choose **Install**. When the feature installation is finished, the Active Directory Domain Services and Active Directory Lightweight Directory Services Tools are available on the Start screen in the **Administrative Tools** folder.

Grant Users and Groups Access to AWS Resources

AWS Directory Service provides the ability to give your directory users and groups access to AWS services and resources, such as access to the Amazon EC2 console. Similar to granting IAM users access to manage directories as described in Identity-Based Policies (IAM Policies) (p. 178), in order for users in your directory to have access to other AWS resources, such as Amazon EC2 you must assign IAM roles and policies to those users and groups. For more information, see IAM Roles in the *IAM User Guide*.

For information about how to grant users access to the AWS Management Console, see Manage Access to the AWS Management Console (p. 145).

Topics

Editing the Trust Relationship for an Existing Role

You can assign your existing IAM roles to your AWS Directory Service users and groups. To do this, however, the role must have a trust relationship with AWS Directory Service. When you use AWS Directory Service to create a role using the procedure in Creating a New Role (p. 153), this trust relationship is automatically set. You only need to establish this trust relationship for IAM roles that are not created by AWS Directory Service.

To establish a trust relationship for an existing role to AWS Directory Service

1. In the navigation pane of the IAM console, choose **Roles**.

   The console displays the roles for your account.
2. Choose the name of the role that you want to modify, and select the **Trust relationships** tab on the details page.
3. Choose **Edit trust relationship**.
4. Under **Policy Document**, paste the following, and then choose **Update Trust Policy**.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Creating a New Role

If you need to create a new IAM role for use with AWS Directory Service, you must create it using the IAM console. For more information, see Creating a Role (AWS Management Console) in the *IAM User Guide*. Once the role has been created, you must then set up a trust relationship with that role before you can see that role in the AWS Directory Service console. For more information, see Editing the Trust Relationship for an Existing Role (p. 153).

Note
The user performing this task must have permission to perform the following IAM actions. For more information, see Identity-Based Policies (IAM Policies) (p. 178).

- iam:PassRole
- iam:GetRole
- iam:CreateRole
- iam:PutRolePolicy

Assigning Users or Groups to an Existing Role

You can assign an existing IAM role to an AWS Directory Service user or group. The role must have a trust relationship with AWS Directory Service. For more information, see Editing the Trust Relationship for an Existing Role (p. 153).

To assign users or groups to an existing IAM role

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Choose the directory ID for your directory.
3. In the **Details** page, choose the **Apps & services** tab.
4. Under **AWS apps & services**, choose **AWS Management Console**.
5. In the **Manage access to AWS Resources** dialog box, choose **Continue**.
6. In the **Assign users and groups to IAM roles** page, choose **click here** to create new IAM roles. If you already have an existing IAM role that has a trust relationship defined in the policy document, skip to the next step.
7. Under **Add Users and Groups to Roles**, choose the link for the existing IAM role that you want to assign users to.
8. In the **Role Detail** page, choose **Add**.
9. In the **Add Users and Groups to Role** page, next to **Select Forest**, choose either the Microsoft AD forest (this forest) or the on-premises forest (trusted forest), whichever contains where the accounts that need access to the AWS Management Console. For more information about how to set up a trusted forest, see Tutorial: Create a Trust Relationship Between Your Microsoft AD and Your On-Premises Domain (p. 73).
10. Next to **Search for**, choose either **User** or **Group**, and then type the name of the user or group. In the list of possible matches, choose the user or group that you want to add.
11. Choose **Add** to finish assigning the users and groups to the role.

Viewing Users and Groups Assigned to a Role

To view the users and groups assigned to a role, perform the following steps.

To view users and group assigned to a role

1. In the AWS Directory Service console navigation pane, choose **Directories**.
2. Choose the directory ID for your directory.
3. In the **Details** page, choose the **Apps & services** tab.
4. Under **AWS apps & services**, choose **AWS Management Console**.
5. In the **Manage access to AWS Resources** dialog box, choose **Continue**.
6. Under **Add Users and Groups to Roles**, choose the role. On the **Role Detail** page, you can view the users and groups assigned to the role.

Removing a User or Group from a Role

To remove a user or group from a role, perform the following steps.

To remove a user or group from a role

1. View the details for the role to remove as explained in Viewing Users and Groups Assigned to a Role (p. 154).
2. In the **Role Detail** page, under **Assigned Users and Groups**, choose the users or groups to remove the role from and choose **Remove**.
3. In the **Remove Role Access** dialog box, confirm that you want to remove the role from the selected users and/or groups, and choose **Remove**. The role is removed from the specified users and groups, but the role is not removed from your account.

Using AWS Managed Policies with AWS Directory Service

AWS Directory Service provides the following AWS managed policies to give your users and groups access to AWS services and resources, such as access to the Amazon EC2 console. You must log in to the AWS Management Console before you can view these policies.

- Read Only Access
- Power User Access
- AWS Directory Service Full Access
- AWS Directory Service Read Only Access
- Amazon Cloud Directory Full Access
- Amazon Cloud Directory Read Only Access
- Amazon EC2 Full Access
- Amazon EC2 Read Only Access
- Amazon VPC Full Access
- Amazon VPC Read Only Access
- Amazon RDS Full Access
- Amazon RDS Read Only Access
- Amazon DynamoDB Full Access
- Amazon DynamoDB Read Only Access
- Amazon S3 Full Access
- Amazon S3 Read Only Access
- AWS CloudTrail Full Access
- AWS CloudTrail Read Only Access
- Amazon CloudWatch Full Access
- Amazon CloudWatch Read Only Access
- Amazon CloudWatch Logs Full Access
- Amazon CloudWatch Logs Read Only Access

For more information on how to create your own policies, see Example Policies for Administering AWS Resources in the *IAM User Guide*.

Add an Instance to Your Directory (Simple AD and Microsoft AD)

You can seamlessly join an EC2 instance to your directory domain when the instance is launched using the Amazon EC2 Systems Manager. For more information, see Seamlessly Joining a Windows Instance to an AWS Directory Service Domain in the *Amazon EC2 User Guide for Windows Instances*.

If you need to manually join an EC2 instance to your domain, you must launch the instance in the proper region and security group or subnet, then join the instance to the domain.

To be able to connect remotely to these instances, you must have IP connectivity to the instances from the network you are connecting from. In most cases, this requires that an Internet gateway be attached to your VPC and that the instance has a public IP address.

Topics

Launching an Instance (Simple AD and Microsoft AD)

To launch an instance to be manually joined to a directory in a VPC

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the region selector in the navigation bar, select the same region as the existing directory.
3. From the Amazon EC2 console dashboard, choose **Launch Instance**.
4. On the **Step 1** page, select the appropriate AMI.
5. On the **Step 2** page, select the appropriate instance type, and then choose **Next**.
6. On the **Step 3** page, do the following, and then choose **Next** :

    1. For **Network**, choose the VPC that your directory was created in.
    2. For **Subnet**, select one of the public subnets in your VPC. The subnet you select must have all external traffic routed to an Internet gateway. If this is not the case, you won't be able to connect to the instance remotely.
    3. For **Auto-assign Public IP**, choose **Enable** (if the subnet setting is not set to enable by default). Although a public IP address is not required to join the domain, in order to connect to the instance, the instance must have a public IP address. Setting this to **Enable** assigns a public IP address automatically, or you can assign an Elastic IP address to the instance after it is launched.
    4. For **Domain join directory**, select your domain from the **Domain join directory** list. To seamlessly join the instance, you also need an IAM role that has the **AmazonEC2RoleforSSM** managed policy attached to it. Select the IAM role that has permission from the IAM role list. If you choose this option, you will not have to manually join the instance to the domain as that will be done for you when the instance is launched.

        Note
        This option is only available for Windows instances. Linux instances must be manually joined to the directory as explained in Manually Add a Linux Instance (Simple AD and Microsoft AD) (p. 158).
    5. For **IAM role**, optionally choose the **Create new IAM role** link to create a new IAM role and attach the AmazonEC2RoleforSSM policy, and then on the **Roles** page, do the following:
        a. Select the **Create role** button.
        b. Under **AWS service**, select the **EC2** link, and then click **Next**.
        c. Under **Select your use case**, select **EC2**, and then select **Next**.
        d. In the list of polices, select the **AmazonEC2RoleforSSM** policy, and then select **Next**.
        e. In **Role name**, type a name for your new role (For example, EC2DomainJoin), in **Role description** type a description (optional), and then select the **Create role** button.
7. Go back to the **Step 3** page, for **IAM role**, choose the refresh icon next to the **IAM role** field. Your new role should be visible in the drop down menu. Select it, and leave the rest of the settings on this page with their default values, and then select **Next**.
8. Continue through the remaining pages in the wizard using your preferred settings.

Note
The security group you select for the instance (on the **Step 6** page) must allow remote access to the instance from your network.

Manually Add a Windows Instance (Simple AD and Microsoft AD)

To manually join an existing Amazon EC2 Windows instance to a Simple AD or AWS Directory Service for Microsoft Active Directory directory, the instance must be launched as specified in Launching an Instance (Simple AD and Microsoft AD) (p. 156).

To join a Windows instance to a Simple AD or Microsoft AD directory

1. Connect to the instance using any Remote Desktop Protocol client.
2. Open the TCP/IPv4 properties dialog box on the instance.

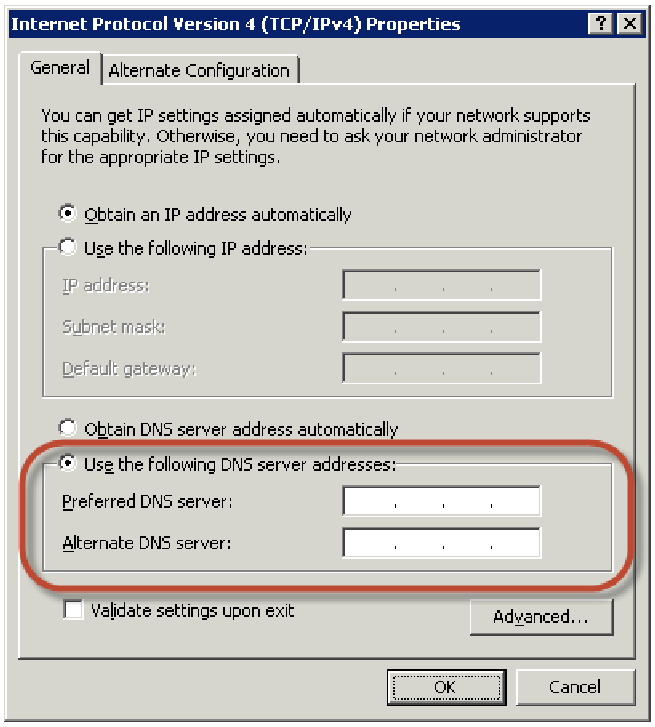
    a. Open **Network Connections**.

    Tip
    You can open **Network Connections** directly by running the following from a command prompt on the instance.

    ```
    %SystemRoot%\system32\control.exe ncpa.cpl
    ```

    b. Open the context menu (right-click) for any enabled network connection and then choose **Properties**.

    c. In the connection properties dialog box, open (double-click) **Internet Protocol Version 4**.
3. (Optional) Select **Use the following DNS server addresses**, change the **Preferred DNS server** and **Alternate DNS server** addresses to the IP addresses of the AWS Directory Service-provided DNS servers, and choose **OK**.


Internet Protocol Version 4 (TCP/IPv4) Properties

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

[✓ Obtain an IP address automatically]
Use the following IP address:
IP address:
Subnet mask:
Default gateway:

Obtain DNS server address automatically
[✓ Use the following DNS server addresses:]
Preferred DNS server:
Alternate DNS server:

Validate settings upon exit

4. Open the **System Properties** dialog box for the instance, select the **Computer Name** tab, and choose **Change**.

   Tip
   You can open the **System Properties** dialog box directly by running the following from a command prompt on the instance.

   ```
   %SystemRoot%\system32\control.exe sysdm.cpl
   ```
5. In the **Member of** field, select **Domain**, enter the fully-qualified name of your AWS Directory Service directory, and choose **OK**.
6. When prompted for the name and password for the domain administrator, enter the username and password of an account that has domain join privileges. For more information about delegating these privileges, see Delegating Directory Join Privileges (Simple AD and Microsoft AD) (p. 162).

   Note
   You can enter either the fully-qualified name of your domain or the NetBios name, followed by a backslash (\), and then the user name, in this case, **Admin**. For example, **corp.example.com\Admin** or **corp\Admin**.
7. After you receive the message welcoming you to the domain, restart the instance to have the changes take effect.

Now that your instance has been joined to the domain, you can log into that instance remotely and install utilities to manage the directory, such as adding users and groups.

Manually Add a Linux Instance (Simple AD and Microsoft AD)

In addition to Amazon EC2 Windows instances, you can also join certain Amazon EC2 Linux instances to a Simple AD or AWS Directory Service for Microsoft Active Directory directory. The following Linux instance distributions and versions are supported:

- Amazon Linux AMI 2015.03
- Red Hat Enterprise Linux 7.2
- Ubuntu Server 14.04 LTS
- CentOS 7

Note
Other Linux distributions and versions may work but have not been tested.

Prerequisites for Joining an Instance to a Simple AD or Microsoft AD Directory

To join a Linux instance to your directory, the instance must be launched as specified in Launching an Instance (Simple AD and Microsoft AD) (p. 156).

Important
When using Simple AD, if you create a user account on a Linux instance with the option "Force user to change password at first login," that user will not be able to initially change their password using **kpasswd**. In order to change the password the first time, a domain administrator must update the user password using the Active Directory Management Tools. Some of the following procedures, if not performed correctly, can render your instance unreachable or unusable. Therefore, we strongly suggest you make a backup or take a snapshot of your instance before performing these procedures.

To join a Linux instance to a Simple AD or Microsoft AD directory

1. Connect to the instance using any SSH client.
2. Configure the Linux instance to use the DNS server IP addresses of the AWS Directory Service-provided DNS servers. You can do this either by setting it up in the DHCP Options set attached to the VPC or by setting it manually on the instance. If you want to set it manually, see How do I assign a static DNS server to a private Amazon EC2 instance in the AWS Knowledge Center for guidance on setting the persistent DNS server for your particular Linux distribution and version.
3. Make sure the instance is up to date.

   Amazon Linux - 64bit/Red Hat - 64bit/CentOS 7

   ```
   $ sudo yum -y update
   ```

   Ubuntu - 64bit

   ```
   $ sudo apt-get update
   $ sudo apt-get -y upgrade
   ```

4. Install the required packages on your Linux instance.

   Note
   Some of these packages may already be installed.
   As you install the packages, particularly in Ubuntu, you might be presented with several pop-up configuration screens. You can generally leave the fields in these screens blank.

   Amazon Linux - 64bit/Red Hat - 64bit

   ```
   $ sudo yum -y install sssd realmd krb5-workstation
   ```

   CentOS 7

   ```
   $ sudo yum -y install sssd realmd krb5-workstation
   $ sudo yum -y install samba-common-tools
   ```

   Ubuntu - 64bit

   ```
   $ sudo apt-get -y install sssd realmd krb5-user samba-common
   ```

5. Join the instance to the directory with the following command.

```
$ sudo realm join -U join_account@example.com example.com --verbose
```

join_account@example.com

An account in the *example.com* domain that has domain join privileges. Enter the password for the account when prompted. For more information about delegating these privileges, see Delegating Directory Join Privileges (Simple AD and Microsoft AD) (p. 162).

example.com

The fully-qualified DNS name of your directory.

```
...
 * Successfully enrolled machine in realm
```

Note
If you are using Ubuntu 14.04, and encounter the following error:
realm: Couldn't join realm: Failed to enroll machine in realm. See diagnostics..
Complete the following additional steps, and then attempt a domain join to resolve the issue:

```
$ killall aptd
$ apt-get install packagekit adcli
```

6. Set the SSH service to allow password authentication.

   a. Open the `/etc/ssh/sshd_config` file in a text editor.

   ```
   sudo vi /etc/ssh/sshd_config
   ```

   b. Set the `PasswordAuthentication` setting to `yes`.

   ```
   PasswordAuthentication yes
   ```

7. Start the SSSD service.

   ```
   $ sudo systemctl start sssd.service
   ```

   Alternatively:

   ```
   $ sudo service sssd start
   ```

8. Restart the instance.
9. After the instance has restarted, connect to it with any SSH client and add the domain admins group to the sudoers list by performing the following steps:

   a. Open the `sudoers` file with the following command:

   ```
   $ sudo vi -f /etc/sudoers
   ```

   b. Do one of the following tasks, depending on which Linux instance distribution you are running:

      i. If you are running Amazon Linux or CentOS, add the following to the bottom of the `sudoers` file and save it.

      ```
      ## Add the "Domain Admins" group from the example.com domain.
      %Domain\ Admins@example.com ALL=(ALL:ALL) ALL
      ```

      (The above example uses "\<space>" to create the Linux space character.)

      ii. If you are running Ubuntu, add the following to the bottom of the `sudoers` file and save it.

      ```
      ## Add the "Domain Admins" group from the example.com domain.
      %Domain\\ Admins@example.com ALL=(ALL:ALL) ALL
      ```

      (The above example uses "\\<space>" with two slashes to create the Linux space character.)

10. By default, all users in the directory can log in to the instance. You can allow only specific users to log in to the instance with **ad_access_filter**. For example:

```
ad_access_filter = (memberOf=cn=admins,ou=Testou,dc=example,dc=com)
```

memberOf

Indicates that users should only be allowed access to the instance if they are a member of a specific group.

cn

The canonical name of the group that should have access. In this example, the group name is *admins*.

ou

This is the organizational unit in which the above group is located. In this example, the OU is *Testou*.

dc

This is the domain component of your domain. In this example, *example*.

dc

This is an additional domain component. In this example, *com*.

You must manually add **ad_access_filter** to your **sssd.conf**. After you do this, your **sssd.conf** might look like this:

```
domains = example.com
config_file_version = 2
services = nss, pam

[domain/example.com]
ad_domain = example.com
krb5_realm = EXAMPLE.COM
realmd_tags = manages-system joined-with-samba
cache_credentials = True
id_provider = ad
krb5_store_password_if_offline = True
default_shell = /bin/bash
ldap_id_mapping = True
use_fully_qualified_names = True
fallback_homedir = /home/%u@%d
access_provider = ad
ad_access_filter = (memberOf=cn=admins,ou=Testou,dc=example,dc=com)
```

Connecting to the Instance

When a user connects to the instance using an SSH client, they are prompted for their username. The user can enter the username in either the `username@example.com` or `EXAMPLE\username` format. The response will appear similar to the following:

```
login as: johndoe@example.com
johndoe@example.com's password:
Last login: Thu Jun 25 16:26:28 2015 from XX.XX.XX.XX
```

Delegating Directory Join Privileges (Simple AD and Microsoft AD)

To join a computer to your directory, you need an account that has privileges to join computers to the directory.

With Simple AD, members of the **Domain Admins** group have sufficient privileges to join computers to the directory.

With AWS Directory Service for Microsoft Active Directory, members of the **Admins** and **Server Admins** groups have these privileges.

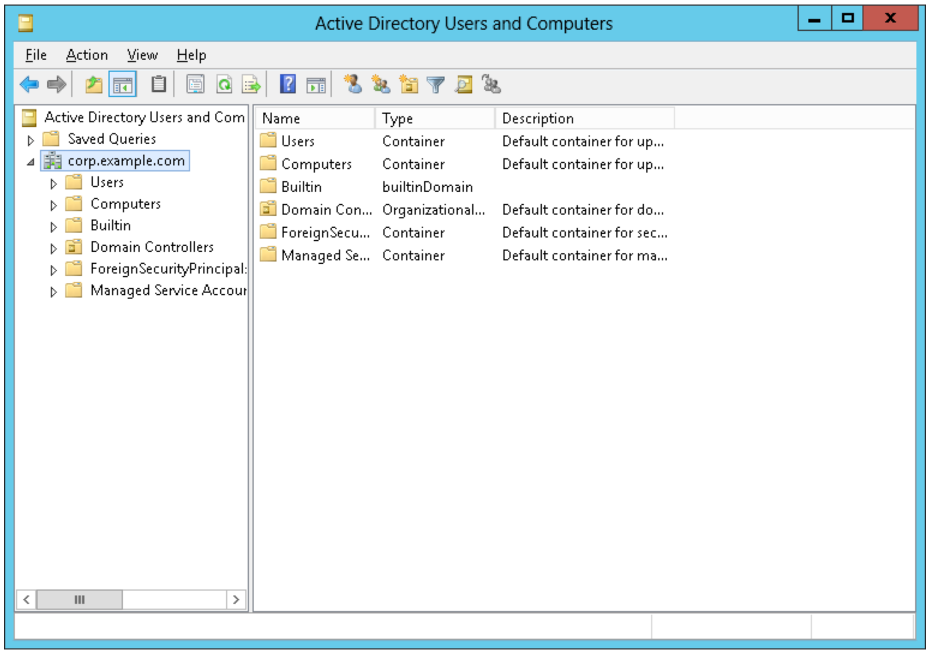
However, as a best practice, you should use an account that has only the minimum privileges necessary. The following procedure demonstrates how to create a new group called `Joiners` and delegate the privileges to this group that are needed to join computers to the directory.

You must perform this procedure on a machine that is joined to your directory and has the **Active Directory User and Computers** MMC snap-in installed. You must also be logged in as a domain administrator.

Simple AD

To delegate join privileges

1. Open **Active Directory User and Computers** and select your domain root in the navigation tree.

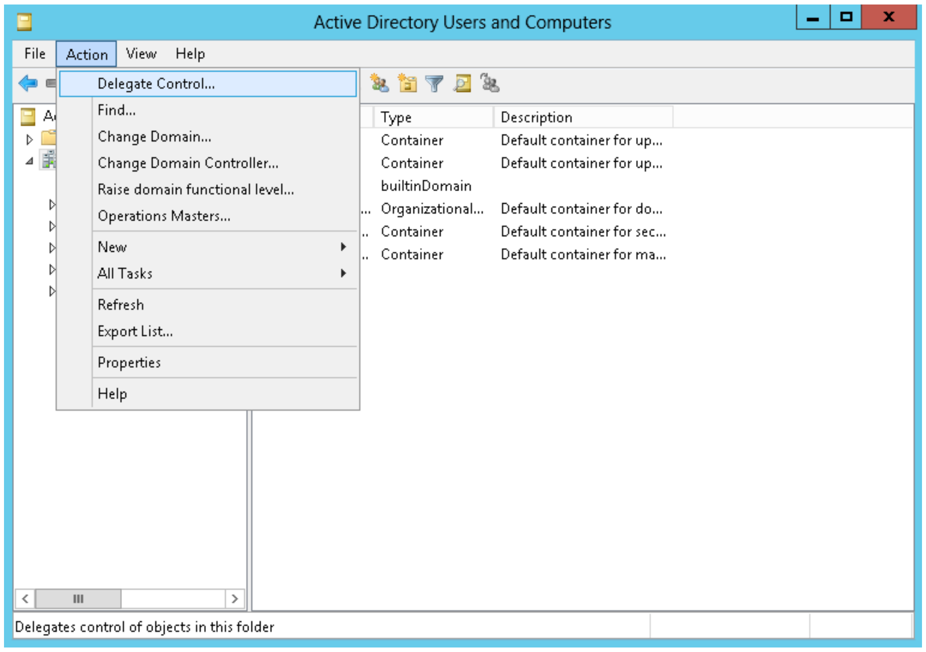


Name	Type	Description
Users	Container	Default container for up...
Computers	Container	Default container for up...
Builtin	builtinDomain	
Domain Con...	Organizational...	Default container for do...
ForeignSecu...	Container	Default container for sec...
Managed Se...	Container	Default container for ma...

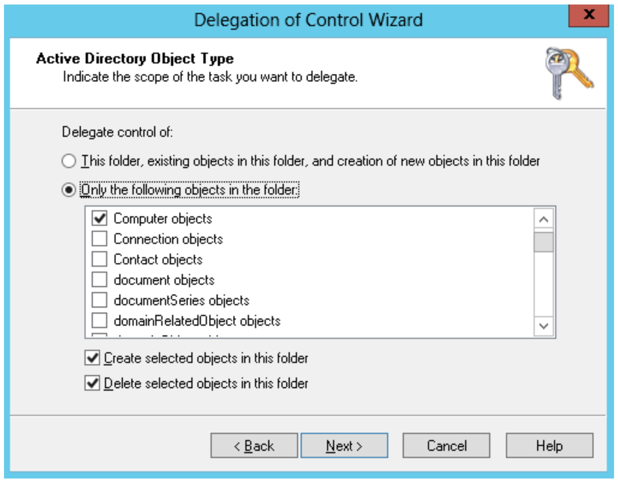
2. In the navigation tree on the left, open the context menu (right-click) for **Users**, choose **New**, and then choose **Group**.
3. In the **New Object - Group** box, type the following and choose **OK**.

   - For **Group name**, type **`Joiners`**.
   - For **Group scope**, choose **Global**.
   - For **Group type**, choose **Security**.

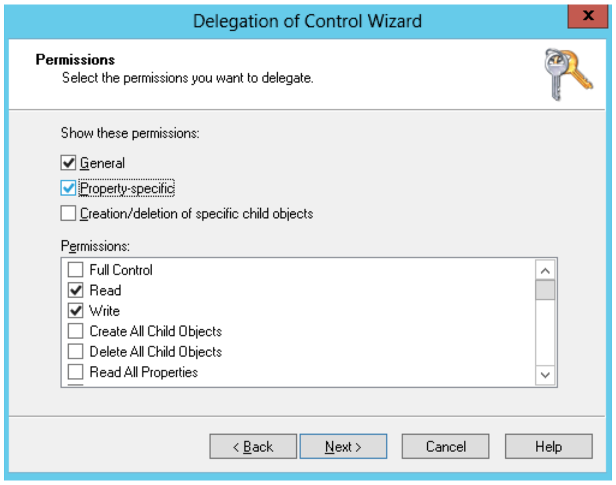
4. In the navigation tree, select your domain root. From the **Action** menu, choose **Delegate Control**.


Active Directory Users and Computers
File
Action
View
Help
Delegate Control...
Find...
Change Domain...
Change Domain Controller...
Raise domain functional level...
Operations Masters...
New
All Tasks
Refresh
Export List...
Properties
Help
Type
Description
Container
Default container for up...
Container
Default container for up...
builtinDomain
Organizational...
Default container for do...
Container
Default container for sec...
Container
Default container for ma...
Delegates control of objects in this folder

5. On the **Delegation of Control Wizard** page, choose **Next**, and then choose **Add**.
6. In the **Select Users, Computers, or Groups** box, type `Joiners` and choose **OK**. If more than one object is found, select the `Joiners` group created above. Choose **Next**.
7. On the **Tasks to Delegate** page, select **Create a custom task to delegate**, and then choose **Next**.
8. Select **Only the following objects in the folder**, and then select **Computer objects**.
9. Select **Create selected objects in this folder** and **Delete selected objects in this folder**. Then choose **Next**.


Delegation of Control Wizard
Active Directory Object Type
Indicate the scope of the task you want to delegate.
Delegate control of:
This folder, existing objects in this folder, and creation of new objects in this folder
Only the following objects in the folder:
☑ Computer objects
☐ Connection objects
☐ Contact objects
☐ document objects
☐ documentSeries objects
☐ domainRelatedObject objects
☑ Create selected objects in this folder
☑ Delete selected objects in this folder
< Back
Next >
Cancel
Help

10. Select **Read** and **Write**, and then choose **Next**.

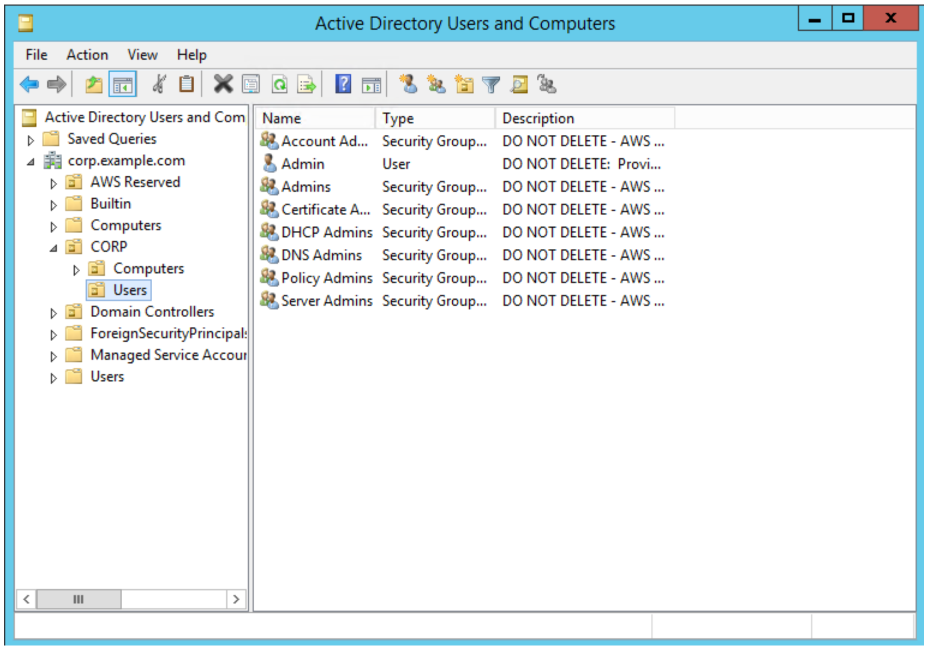


11. Verify the information on the **Completing the Delegation of Control Wizard** page and choose **Finish**.
12. Create a user with a strong password and add that user to the `Joiners` group. The user will then have sufficient privileges to connect AWS Directory Service to the directory.

AWS Directory Service for Microsoft Active Directory

To delegate join privileges

1. Open **Active Directory User and Computers** and select the organizational unit (OU) that has your NetBIOS name in the navigation tree, then select the **Users** OU.



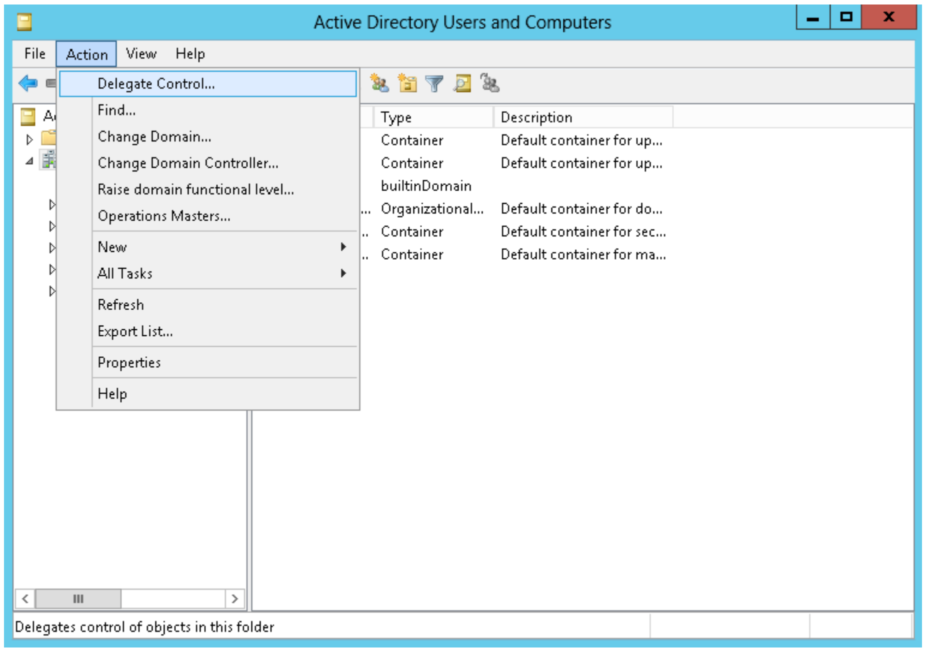
Name	Type	Description
Account Ad...	Security Group...	DO NOT DELETE - AWS ...
Admin	User	DO NOT DELETE: Provi...
Admins	Security Group...	DO NOT DELETE - AWS ...
Certificate A...	Security Group...	DO NOT DELETE - AWS ...
DHCP Admins	Security Group...	DO NOT DELETE - AWS ...
DNS Admins	Security Group...	DO NOT DELETE - AWS ...
Policy Admins	Security Group...	DO NOT DELETE - AWS ...
Server Admins	Security Group...	DO NOT DELETE - AWS ...

Important
When you launch a AWS Directory Service for Microsoft Active Directory, AWS creates an organizational unit (OU) that contains all your directory's objects. This OU, which has the NetBIOS name that you typed when you created your directory, is located in the domain root. The domain root is owned and managed by AWS. You cannot make changes to the

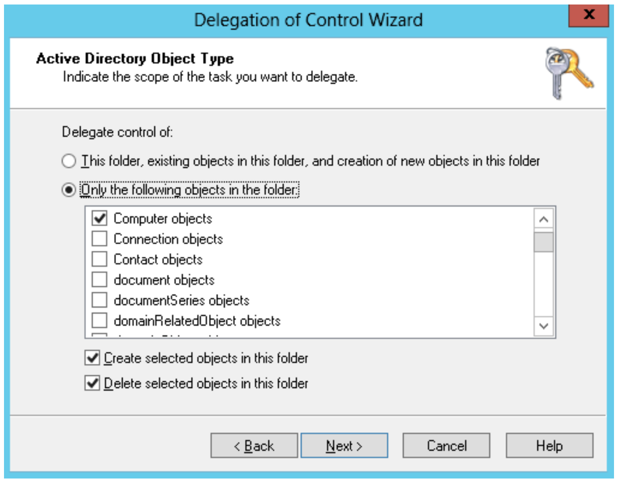
domain root itself, therefore, you must create the **`Joiners`** group within the OU that has your NetBIOS name.

2. Open the context menu (right-click) for **Users**, choose **New**, and then choose **Group**.
3. In the **New Object - Group** box, type the following and choose **OK**.

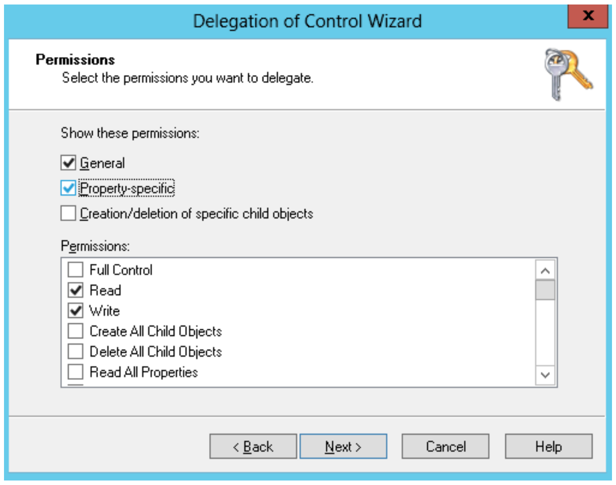
   - For **Group name**, type **`Joiners`**.
   - For **Group scope**, choose **Global**.
   - For **Group type**, choose **Security**.
4. In the navigation tree, select the **Computers** container under your NetBIOS name. From the **Action** menu, choose **Delegate Control**.


Active Directory Users and Computers
File
Action
View
Help
Delegate Control...
Find...
Change Domain...
Change Domain Controller...
Raise domain functional level...
Operations Masters...
New
All Tasks
Refresh
Export List...
Properties
Help
Type
Description
Container
Default container for up...
Container
Default container for up...
builtinDomain
Organizational...
Default container for do...
Container
Default container for sec...
Container
Default container for ma...
Delegates control of objects in this folder

5. On the **Delegation of Control Wizard** page, choose **Next**, and then choose **Add**.
6. In the **Select Users, Computers, or Groups** box, type `Joiners` and choose **OK**. If more than one object is found, select the `Joiners` group created above. Choose **Next**.
7. On the **Tasks to Delegate** page, select **Create a custom task to delegate**, and then choose **Next**.
8. Select **Only the following objects in the folder**, and then select **Computer objects**.
9. Select **Create selected objects in this folder** and **Delete selected objects in this folder**. Then choose **Next**.


Delegation of Control Wizard
Active Directory Object Type
Indicate the scope of the task you want to delegate.
Delegate control of:
This folder, existing objects in this folder, and creation of new objects in this folder
Only the following objects in the folder:
Computer objects
Connection objects
Contact objects
document objects
documentSeries objects
domainRelatedObject objects
Create selected objects in this folder
Delete selected objects in this folder
< Back
Next >
Cancel
Help

10. Select **Read** and **Write**, and then choose **Next**.



11. Verify the information on the **Completing the Delegation of Control Wizard** page and choose **Finish**.
12. Create a user with a strong password and add that user to the `Joiners` group. This user must be in the **Users** container that is under your NetBIOS name. The user will then have sufficient privileges to connect instances to the directory.

Using DNS with Simple AD and Microsoft AD

DNS and Simple AD

Simple AD forwards DNS requests to the IP address of the Amazon-provided DNS servers for your VPC. These DNS servers will resolve names configured in your Amazon Route 53 private hosted zones. By pointing your on-premises computers to your Simple AD, you can now resolve DNS requests to the private hosted zone.

Note that to enable your Simple AD to respond to external DNS queries, the network access control list (ACL) for the VPC containing your Simple AD must be configured to allow traffic from outside the VPC.

If you are not using Amazon Route 53 private hosted zones, your DNS requests will be forwarded to public DNS servers.

If you're using custom DNS servers that are outside of your VPC and you want to use private DNS, you must reconfigure to use custom DNS servers on EC2 instances within your VPC. For more information, see Working with Private Hosted Zones.

If you want your Simple AD to resolve names using both DNS servers within your VPC and private DNS servers outside of your VPC, you can do this using a DHCP options set. For a detailed example, see this article.

> **Note**
> DNS dynamic updates are not supported in Simple AD domains. You can instead make the changes directly by connecting to your directory using DNS Manager on an instance that is joined to your domain.

For more information on Amazon Route 53, see What is Amazon Route 53.

DNS and AWS Directory Service for Microsoft Active Directory

The Microsoft AD directory service is tightly integrated with Microsoft Active Directory. Members of either the **Admins** group or the **DNS Admins** group can manage DNS zones, records, logs, forwarders and more using a variety of tools, such as the DNSMGMT console, PowerShell, or the DNSCMD features included with Microsoft's Remote Server Administration Tools (RSAT).

DHCP Options Set

AWS recommends that you create a DHCP options set for your AWS Directory Service directory and assign the DHCP options set to the VPC that your directory is in. This allows any instances in that VPC to point to the specified domain and DNS servers to resolve their domain names.

For more information about DHCP options sets, see DHCP Options Sets in the *Amazon VPC User Guide*.

To create a DHCP options set for your directory

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose **DHCP Options Sets** in the navigation pane, and choose **Create DHCP options set**.
3. In the **Create DHCP options set** dialog box, enter the following values for your directory:

   Name tag

      An optional tag for the options set.

   Domain name

      The fully-qualified name of your directory, such as `corp.example.com`.

   Domain name servers

      The IP addresses of your directory's DNS servers. These are the IP addresses of your AWS-provided directory. You can find these addresses by going to the AWS Directory Service console navigation pane, selecting **Directories** and then choosing the correct directory ID.

   NTP servers

      Leave this field blank.

   NetBIOS name servers

      Leave this field blank.

   NetBIOS node type

      Leave this field blank.

4. Choose **Yes, Create**. The new set of DHCP options appears in your list of DHCP options.
5. Make a note of the ID of the new set of DHCP options (dopt-*xxxxxxxx*). You need it to associate the new options set with your VPC.

To change the DHCP options set associated with a VPC

After you create a set of DHCP options, you can't modify them. If you want your VPC to use a different set of DHCP options, you must create a new set and associate them with your VPC. You can also set up your VPC to use no DHCP options at all.

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose **Your VPCs** in the navigation pane.
3. Select the VPC, and choose **Edit DHCP Options Set** from the **Actions** list.
4. In the **DHCP Options Set** list, select the desired options set from the list, and choose **Save**.

Single Sign-On

AWS Directory Service provides the ability to allow your users to access Amazon WorkDocs from a computer joined to the directory without having to enter their credentials separately.

Before you enable single sign-on, you need to take additional steps to enable your users' web browsers to support single sign-on. Users may need to modify their web browser settings to enable single sign-on. For more information, see Single Sign-On for IE and Chrome (p. 169) and Single Sign-On for Firefox (p. 173).

> **Note**
> Single sign-on only works when used on a computer that is joined to the AWS Directory Service directory. It cannot be used on computers that are not joined to the directory.

To enable or disable single sign-on with Amazon WorkDocs

1. In the AWS Directory Service console navigation pane, select **Directories**.
2. Choose the directory ID link for your directory.
3. Choose the **Apps & Services** tab.
4. Under **Access URL**, choose **Enable** to enable single sign-on for Amazon WorkDocs.

   If you do not see the **Enable** button, you may need to first create an Access URL before this option will be displayed. For more information about how to create an access URL, see Creating an Access URL (p. 144).
5. In the **Enable Single Sign-On for this directory** dialog box, choose **Enable**. Single sign-on is enabled for the directory.

   If the directory is an AD Connector directory and the AD Connector service account does not have permission to add a service principal name, you are prompted for the username and password for a directory user that has this permission. These credentials are only used to enable single sign-on and are not stored by the service. The AD Connector service account is not changed.
6. If you later want to disable single sign-on with Amazon WorkDocs, choose **Disable**, and then in the **Disable Single Sign-On for this directory** dialog box, choose **Disable** again.

   If the directory is an AD Connector directory and the AD Connector service account does not have permission to remove a service principal name, you are prompted for the username and password for a directory user that has this permission. These credentials are only used to disable single sign-on and are not stored by the service. The AD Connector service account is not changed.

Topics

Single Sign-On for IE and Chrome

To allow Microsoft's Internet Explorer (IE) and Google's Chrome browsers to support single sign-on, the following tasks must be performed on the client computer:

- Add your access URL (e.g., https://*<alias>*.awsapps.com) to the list of approved sites for single sign-on.
- Enable active scripting (JavaScript).
- Allow automatic logon.
- Enable integrated authentication.

You or your users can perform these tasks manually, or you can change these settings using Group Policy settings.

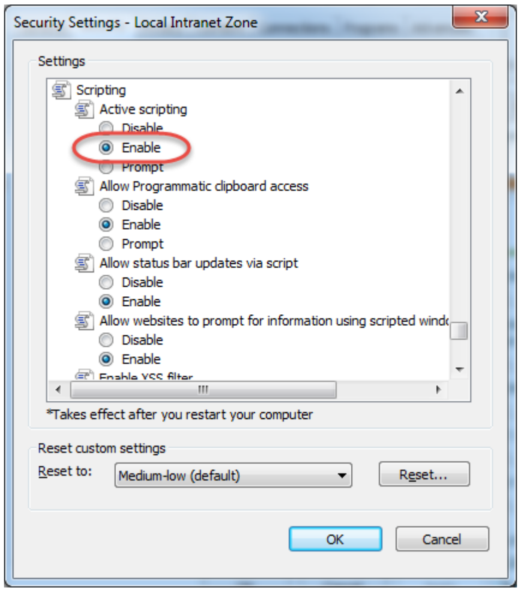
Topics

Manual Update for Single Sign-On on Windows

To manually enable single sign-on on a Windows computer, perform the following steps on the client computer. Some of these settings may already be set correctly.

To manually enable single sign-on for Internet Explorer and Chrome on Windows

1. To open the **Internet Properties** dialog box, choose the **Start** menu, type `Internet Options` in the search box, and choose **Internet Options**.
2. Add your access URL to the list of approved sites for single sign-on by performing the following steps:
   a. In the **Internet Properties** dialog box, select the **Security** tab.
   b. Select **Local intranet** and choose **Sites**.
   c. In the **Local intranet** dialog box, choose **Advanced**.
   d. Add your access URL to the list of websites and choose **Close**.
   e. In the **Local intranet** dialog box, choose **OK**.
3. To enable active scripting, perform the following steps:
   a. In the **Security** tab of the **Internet Properties** dialog box, choose **Custom level**.
   b. In the **Security Settings - Local Intranet Zone** dialog box, scroll down to **Scripting** and select **Enable** under **Active scripting**.



Security Settings - Local Intranet Zone

Settings

- Scripting
  - Active scripting
    - Disable
    - [✓ Enable]
    - Prompt
  - Allow Programmatic clipboard access
    - Disable
    - [✓ Enable]
    - Prompt
  - Allow status bar updates via script
    - Disable
    - [✓ Enable]
  - Allow websites to prompt for information using scripted windo
    - Disable
    - [✓ Enable]
  - Enable XSS filter

*Takes effect after you restart your computer

Reset custom settings
Reset to: Medium-low (default) Reset...

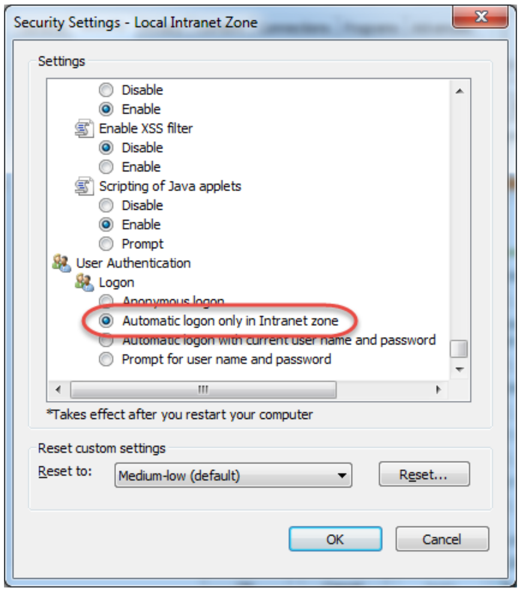
OK Cancel

c. In the **Security Settings - Local Intranet Zone** dialog box, choose **OK**.

4. To enable automatic logon, perform the following steps:

   a. In the **Security** tab of the **Internet Properties** dialog box, choose **Custom level**.

   b. In the **Security Settings - Local Intranet Zone** dialog box, scroll down to **User Authentication** and select **Automatic logon only in Intranet zone** under **Logon**.



Security Settings - Local Intranet Zone

Settings

- Disable
- [✓ Enable]
- Enable XSS filter
  - [✓ Disable]
  - Enable
- Scripting of Java applets
  - Disable
  - [✓ Enable]
  - Prompt
- User Authentication
  - Logon
    - Anonymous logon
    - [✓ Automatic logon only in Intranet zone]
    - Automatic logon with current user name and password
    - Prompt for user name and password

*Takes effect after you restart your computer

Reset custom settings
Reset to: Medium-low (default) Reset...

OK Cancel

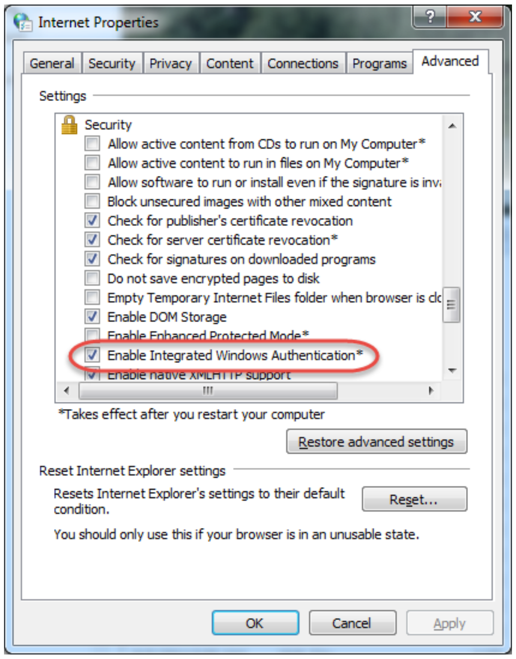
   c. In the **Security Settings - Local Intranet Zone** dialog box, choose **OK**.

   d. In the **Security Settings - Local Intranet Zone** dialog box, choose **OK**.

5. To enable integrated authentication, perform the following steps:

   a. In the **Internet Properties** dialog box, select the **Advanced** tab.

   b. Scroll down to **Security** and select **Enable Integrated Windows Authentication**.



Internet Properties

General | Security | Privacy | Content | Connections | Programs | Advanced

Settings

Security
- [ ] Allow active content from CDs to run on My Computer*
- [ ] Allow active content to run in files on My Computer*
- [ ] Allow software to run or install even if the signature is inv
- [ ] Block unsecured images with other mixed content
- [✓] Check for publisher's certificate revocation
- [✓] Check for server certificate revocation*
- [✓] Check for signatures on downloaded programs
- [ ] Do not save encrypted pages to disk
- [ ] Empty Temporary Internet Files folder when browser is cl
- [✓] Enable DOM Storage
- [ ] Enable Enhanced Protected Mode*
- [✓] Enable Integrated Windows Authentication*
- [✓] Enable native XMLHTTP support

*Takes effect after you restart your computer

Restore advanced settings

Reset Internet Explorer settings

Resets Internet Explorer's settings to their default condition.

Reset...

You should only use this if your browser is in an unusable state.

OK | Cancel | Apply

c. In the **Internet Properties** dialog box, choose **OK**.

6. Close and re-open your browser to have these changes take effect.

Manual Update for Single Sign-On on OS X

To manually enable single sign-on for Chrome on OS X, perform the following steps on the client computer. You will need administrator rights on your computer to complete these steps.

To manually enable single sign-on for Chrome on OS X

1. Add your access URL to the AuthServerWhitelist policy by running the following command:

```
defaults write com.google.Chrome AuthServerWhitelist "https://<alias>.awsapps.com"
```

2. Open **System Preferences**, go to the **Profiles** panel, and delete the `Chrome Kerberos Configuration` profile.
3. Restart Chrome and open chrome://policy in Chrome to confirm that the new settings are in place.

Group Policy Settings for Single Sign-On

The domain administrator can implement Group Policy settings to make the single sign-on changes on client computers that are joined to the domain.

Note
If you manage the Chrome web browsers on the computers in your domain with Chrome policies, you must add your access URL to the AuthServerWhitelist policy. For more information about setting Chrome policies, go to Policy Settings in Chrome.

To enable single sign-on for Internet Explorer and Chrome using Group Policy settings

1. Create a new Group Policy object by performing the following steps:

   a. Open the Group Policy Management tool, navigate to your domain and select **Group Policy Objects**.
   b. From the main menu, choose **Action** and select **New**.
   c. In the **New GPO** dialog box, enter a descriptive name for the Group Policy object, such as `SSO Policy`, and leave **Source Starter GPO** set to **(none)**. Click **OK**.

2. Add the access URL to the list of approved sites for single sign-on by performing the following steps:

   a. In the Group Policy Management tool, navigate to your domain, select **Group Policy Objects**, open the context (right-click) menu for your SSO policy, and choose **Edit**.
   b. In the policy tree, navigate to **User Configuration** > **Preferences** > **Windows Settings**.
   c. In the **Windows Settings** list, open the context (right-click) menu for **Registry** and choose **New registry item**.
   d. In the **New Registry Properties** dialog box, enter the following settings and choose **OK**:

      Action

         `Update`

      Hive

         `HKEY_CURRENT_USER`

      Path

         `Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains\awsapps.com\<alias>`

         The value for *`<alias>`* is derived from your access URL. If your access URL is `https://examplecorp.awsapps.com`, the alias is `examplecorp`, and the registry key will be `Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains\awsapps.com\examplecorp`.

      Value name

         `https`

      Value type

         `REG_DWORD`

      Value data

         `1`

3. To enable active scripting, perform the following steps:

   a. In the Group Policy Management tool, navigate to your domain, select **Group Policy Objects**, open the context (right-click) menu for your SSO policy, and choose **Edit**.
   b. In the policy tree, navigate to **Computer Configuration** > **Policies** > **Administrative Templates** > **Windows Components** > **Internet Explorer** > **Internet Control Panel** > **Security Page** > **Intranet Zone**.
   c. In the **Intranet Zone** list, open the context (right-click) menu for **Allow active scripting** and choose **Edit**.
   d. In the **Allow active scripting** dialog box, enter the following settings and choose **OK**:

      - Select the **Enabled** radio button.
      - Under **Options** set **Allow active scripting** to **Enable**.

4. To enable automatic logon, perform the following steps:

   a. In the Group Policy Management tool, navigate to your domain, select Group Policy Objects, open the context (right-click) menu for your SSO policy, and choose **Edit**.
   b. In the policy tree, navigate to **Computer Configuration** > **Policies** > **Administrative Templates** > **Windows Components** > **Internet Explorer** > **Internet Control Panel** > **Security Page** > **Intranet Zone**.
   c. In the **Intranet Zone** list, open the context (right-click) menu for **Logon options** and choose **Edit**.
   d. In the **Logon options** dialog box, enter the following settings and choose **OK**:
      - Select the **Enabled** radio button.
      - Under **Options** set **Logon options** to **Automatic logon only in Intranet zone**.

5. To enable integrated authentication, perform the following steps:
   a. In the Group Policy Management tool, navigate to your domain, select **Group Policy Objects**, open the context (right-click) menu for your SSO policy, and choose **Edit**.
   b. In the policy tree, navigate to **User Configuration** > **Preferences** > **Windows Settings**.
   c. In the **Windows Settings** list, open the context (right-click) menu for **Registry** and choose **New registry item**.
   d. In the **New Registry Properties** dialog box, enter the following settings and choose **OK**:

      Action

      `Update`

      Hive

      `HKEY_CURRENT_USER`

      Path

      `Software\Microsoft\Windows\CurrentVersion\Internet Settings`

      Value name

      `EnableNegotiate`

      Value type

      `REG_DWORD`

      Value data

      `1`
6. Close the **Group Policy Management Editor** window if it is still open.
7. Assign the new policy to your domain by following these steps:
   a. In the Group Policy Management tree, open the context (right-click) menu for your domain and choose **Link an Existing GPO**.
   b. In the **Group Policy Objects** list, select your SSO policy and choose **OK**.

These changes will take effect after the next Group Policy update on the client, or the next time the user logs in.

Single Sign-On for Firefox

To allow Mozilla's Firefox browser to support single sign-on, add your access URL (e.g., https://*<alias>*.awsapps.com) to the list of approved sites for single sign-on. This can be done manually, or automated with a script.

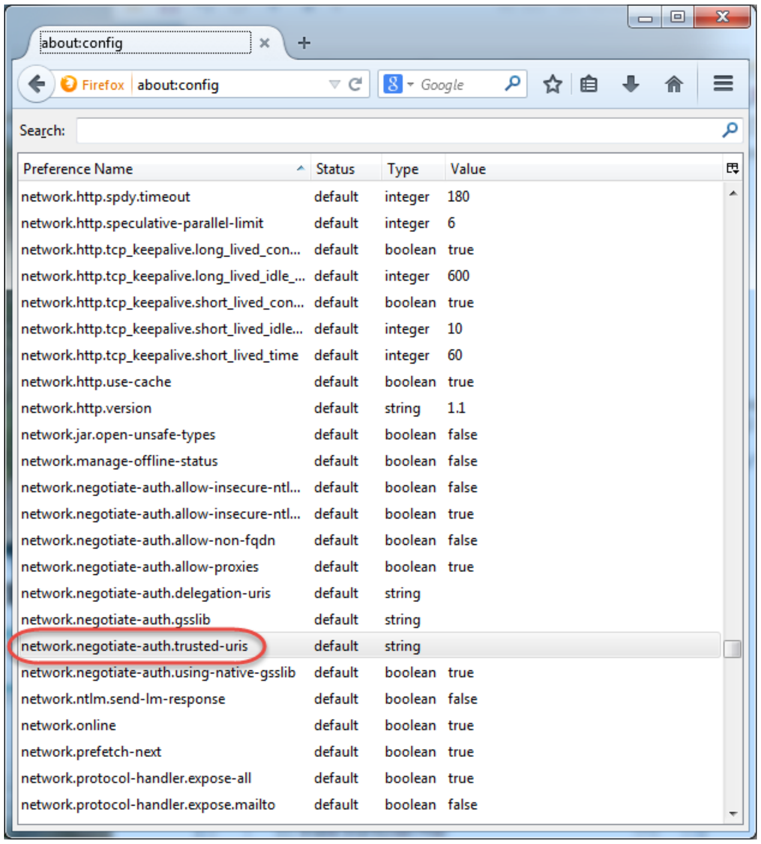
Topics

Manual Update for Single Sign-On

To manually add your access URL to the list of approved sites in Firefox, perform the following steps on the client computer.

To manually add your access URL to the list of approved sites in Firefox

1. Open Firefox and open the `about:config` page.
2. Open the `network.negotiate-auth.trusted-uris` preference and add your access URL to the list of sites. Use a comma (,) to separate multiple entries.



Preference Name	Status	Type	Value
network.http.spdy.timeout	default	integer	180
network.http.speculative-parallel-limit	default	integer	6
network.http.tcp_keepalive.long_lived_con...	default	boolean	true
network.http.tcp_keepalive.long_lived_idle_...	default	integer	600
network.http.tcp_keepalive.short_lived_con...	default	boolean	true
network.http.tcp_keepalive.short_lived_idle...	default	integer	10
network.http.tcp_keepalive.short_lived_time	default	integer	60
network.http.use-cache	default	boolean	true
network.http.version	default	string	1.1
network.jar.open-unsafe-types	default	boolean	false
network.manage-offline-status	default	boolean	false
network.negotiate-auth.allow-insecure-ntl...	default	boolean	false
network.negotiate-auth.allow-insecure-ntl...	default	boolean	true
network.negotiate-auth.allow-non-fqdn	default	boolean	false
network.negotiate-auth.allow-proxies	default	boolean	true
network.negotiate-auth.delegation-uris	default	string	
network.negotiate-auth.gsslib	default	string	
network.negotiate-auth.trusted-uris	default	string	
network.negotiate-auth.using-native-gsslib	default	boolean	true
network.ntlm.send-lm-response	default	boolean	false
network.online	default	boolean	true
network.prefetch-next	default	boolean	true
network.protocol-handler.expose-all	default	boolean	true
network.protocol-handler.expose.mailto	default	boolean	false

Automatic Update for Single Sign-On

As a domain administrator, you can use a script to add your access URL to the Firefox `network.negotiate-auth.trusted-uris` user preference on all computers on your network. For more information, go to https://support.mozilla.org/en-US/questions/939037.

Authentication and Access Control for AWS Directory Service

Access to AWS Directory Service requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an Amazon Cloud Directory (p. 11) or other AWS Directory Service directory. The following sections provide details on how you can use AWS Identity and Access Management (IAM) and AWS Directory Service to help secure your resources by controlling who can access them:

- Authentication (p. 175)
- Access Control (p. 176)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.
- **IAM user** – An IAM user is an identity within your AWS account that has specific custom permissions (for example, permissions to create a directory in AWS Directory Service). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

  In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (CLI). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. AWS Directory Service supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *AWS General Reference*.

- **IAM role** – An IAM role is an IAM identity that you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:

  - **Federated user access** – Instead of creating an IAM user, you can use existing user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the *IAM User Guide*.

- **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access AWS Directory Service resources. For example, you must have permissions to create an Amazon Cloud Directory or other AWS Directory Service directory or to create a directory snapshot.

The following sections describe how to manage permissions for AWS Directory Service. We recommend that you read the overview first.

Overview of Managing Access Permissions to Your AWS Directory Service Resources

Every AWS resource is owned by an AWS account, and permissions to create or access the resources are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

Note
An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

AWS Directory Service Resources and Operations

In AWS Directory Service, the primary resource is a *directory*. AWS Directory Service supports directory snapshot resources as well. However, you can create snapshots only in the context of an existing directory. Therefore, a snapshot is referred to as a *subresource*.

These resources have unique Amazon Resource Names (ARNs) associated with them as shown in the following table.

Resource Type	ARN Format
Directory	`arn:aws:ds:region:account-id:directory/external-directory-id`
Snapshot	`arn:aws:ds:region:account-id:snapshot/external-snapshot-id`

AWS Directory Service provides a set of operations to work with the appropriate resources. For a list of available operations, see either Amazon Cloud Directory Actions or Directory Service Actions.

Understanding Resource Ownership

A *resource owner* is the AWS account that created a resource. That is, the resource owner is the AWS account of the *principal entity* (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create an AWS Directory Service resource, such as a directory, your AWS account is the owner of that resource.
- If you create an IAM user in your AWS account and grant permissions to create AWS Directory Service resources to that user, the user can also create AWS Directory Service resources. However, your AWS account, to which the user belongs, owns the resources.
- If you create an IAM role in your AWS account with permissions to create AWS Directory Service resources, anyone who can assume the role can create AWS Directory Service resources. Your AWS account, to which the role belongs, owns the AWS Directory Service resources.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

> **Note**
> This section discusses using IAM in the context of AWS Directory Service. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is IAM? in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM polices) and policies attached to a resource are referred to as *resource-based* policies. AWS Directory Service supports only identity-based policies (IAM policies).

Topics

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – An account administrator can use a permissions policy that is associated with a particular user to grant permissions for that user to create an AWS Directory Service resource, such as a new directory.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:
  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
  2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

  For more information about using IAM to delegate permissions, see Access Management in the *IAM User Guide*.

The following permissions policy grants permissions to a user to run all of the actions that begin with `Describe`. These actions show information about an AWS Directory Service resource, such as a directory or snapshot. Note that the wildcard character (*) in the `Resource` element indicates that the actions are allowed for all AWS Directory Service resources owned by the account.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"ds:Describe*",
         "Resource":"*"
      }
   ]
}
```

For more information about using identity-based policies with AWS Directory Service, see Using Identity-Based Policies (IAM Policies) for AWS Directory Service (p. 179). For more information about users, groups, roles, and permissions, see Identities (Users, Groups, and Roles) in the *IAM User Guide*.

Resource-Based Policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. AWS Directory Service doesn't support resource-based policies.

Specifying Policy Elements: Actions, Effects, Resources, and Principals

For each AWS Directory Service resource (see AWS Directory Service Resources and Operations (p. 177)), the service defines a set of API operations. For a list of available API operations, see either Amazon Cloud Directory Actions or Directory Service Actions. To grant permissions for these API operations, AWS Directory Service defines a set of actions that you can specify in a policy. Note that, performing an API operation can require permissions for more than one action.

The following are the basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For AWS Directory Service resources, you always use the wildcard character (*) in IAM policies. For more information, see AWS Directory Service Resources and Operations (p. 177).
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, the `ds:DescribeDirectories` permission allows the user permissions to perform the AWS Directory Service `DescribeDirectories` operation.
- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). AWS Directory Service doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

For a table showing all of the Amazon Cloud Directory or AWS Directory Service API actions and the resources that they apply to, see Amazon Cloud Directory API Permissions: Actions, Resources, and Conditions Reference (p. 183) or AWS Directory Service API Permissions: Actions, Resources, and Conditions Reference (p. 183).

Specifying Conditions in a Policy

When you grant permissions, you can use the access policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see Condition in the *IAM User Guide*.

To express conditions, you use predefined condition keys. There are no condition keys specific to AWS Directory Service. However, there are AWS-wide condition keys that you can use as appropriate. For a complete list of AWS-wide keys, see Available Keys for Conditions in the *IAM User Guide*.

Using Identity-Based Policies (IAM Policies) for AWS Directory Service

This topic provides examples of identity-based policies in which an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

> **Important**
> We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your AWS Directory Service resources. For more information, see Overview of Managing Access Permissions to Your AWS Directory Service Resources (p. 176).

The sections in this topic cover the following:

- Permissions Required to Use the AWS Directory Service Console (p. 181)
- AWS Managed (Predefined) Policies for Amazon Cloud Directory and AWS Directory Service (p. 181)
- Customer Managed Policy Examples (p. 182)

The following shows an example of a permissions policy.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Action" : [
        "ds:CreateDirectory"
      ],
      "Effect" : "Allow",
      "Resource" : "*"
    },

    {
      "Action" : [
        "iam:PassRole",
        "iam:GetRole",
        "iam:CreateRole",
        "iam:PutRolePolicy"
       ],
      "Effect" : "Allow",
      "Resource" : "*"
    },

    {
      "Action" : [
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:CreateSecurityGroup",
        "ec2:DeleteSecurityGroup",
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupEgress"
      ],
      "Effect" : "Allow",
      "Resource" : "*"
    }
  ]
}
```

The policy includes the following:

- The first statement grants permission to create a AWS Directory Service directory. AWS Directory Service doesn't support permissions for this particular action at the resource-level. Therefore, the policy specifies a wildcard character (*) as the `Resource` value.
- The second statement grants permissions to certain IAM actions. The access to IAM actions is needed so that AWS Directory Service can read and create IAM roles on your behalf. The wildcard character (*) at the end of the `Resource` value means that the statement allows permission for the IAM actions on any IAM role. To limit this permission to a specific role, replace the wildcard character (*) in the resource ARN with the specific role name. For more information, see IAM Actions.
- The third statement grants permissions to a specific set of Amazon EC2 resources that are necessary to allow AWS Directory Service to create, configure, and destroy its directories. The wildcard character

(*) at the end of the `Resource` value means that the statement allows permission for the EC2 actions on any EC2 resource or subresource. To limit this permission to a specific role, replace the wildcard character (*) in the resource ARN with the specific resource or subresource. For more information, see EC2 Actions

The policy doesn't specify the `Principal` element because in an identity-based policy you don't specify the principal who gets the permission. When you attach policy to a user, the user is the implicit principal. When you attach a permission policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the Amazon Cloud Directory or AWS Directory Service API actions and the resources that they apply to, see Amazon Cloud Directory API Permissions: Actions, Resources, and Conditions Reference (p. 183) or AWS Directory Service API Permissions: Actions, Resources, and Conditions Reference (p. 183).

Permissions Required to Use the AWS Directory Service Console

For a user to work with the AWS Directory Service console, that user must have permissions listed in the policy above or the permissions granted by the Directory Service Full Access Role or Directory Service Read Only role, described in AWS Managed (Predefined) Policies for Amazon Cloud Directory and AWS Directory Service (p. 181).

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended for users with that IAM policy.

AWS Managed (Predefined) Policies for Amazon Cloud Directory and AWS Directory Service

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see AWS Managed Policies in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon Cloud Directory:

- **AmazonCloudDirectoryReadOnlyAccess** – Grants a user or group read-only access to all Amazon Cloud Directory resources. For more information, see Using AWS Managed Policies with AWS Directory Service (p. 155).
- **AmazonCloudDirectoryFullAccess** – Grants a user or group full access to Amazon Cloud Directory. For more information, see Using AWS Managed Policies with AWS Directory Service (p. 155).

The following AWS managed policies, which you can attach to users in your account, are specific to AWS Directory Service:

- **AWSDirectoryServiceReadOnlyAccess** – Grants a user or group read-only access to all AWS Directory Service resources, EC2 subnets, EC2 network interfaces, and AWS SNS topics and subscriptions for the root AWS account. For more information, see Using AWS Managed Policies with AWS Directory Service (p. 155).
- **AWSDirectoryServiceFullAccess** – Grants a user or group the following:
  - Full access to AWS Directory Service
  - Access to key Amazon EC2 services required to use AWS Directory Service

- Ability to list Amazon Simple Notification Service topics
- Ability to create, manage, and delete Amazon Simple Notification Service topics with a name beginning with “DirectoryMonitoring”

For more information, see Using AWS Managed Policies with AWS Directory Service (p. 155).

In addition, there are other AWS-managed policies that are suitable for use with other IAM roles. These policies are assigned to the roles associated with users in your Amazon Cloud Directory or other AWS Directory Service directory and are required in order for those users to have access to other AWS resources, such as Amazon EC2. For more information, see Grant Users and Groups Access to AWS Resources (p. 152).

You can also create custom IAM policies that allow users to access the required API actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various AWS Directory Service actions.

Note
All examples use the US West (Oregon) Region (`us-west-2`) and contain fictitious account IDs.

Examples

Example 1: Allow a User to Perform Any Describe Action on Any AWS Directory Service Resource

The following permissions policy grants permissions to a user to run all of the actions that begin with `Describe`. These actions show information about an AWS Directory Service resource, such as a directory or snapshot. Note that the wildcard character (*) in the `Resource` element indicates that the actions are allowed for all AWS Directory Service resources owned by the account.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"ds:Describe*",
         "Resource":"*"
      }
   ]
}
```

Example 2: Allow a User to Create a Directory

The following permissions policy grants permissions to allow a user to create a directory and all other related resources, such as snapshots and trusts. In order to do so, permissions to certain Amazon EC2 services are also required.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": [
                "ds:Create*",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateNetworkInterface",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteSecurityGroup",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
                  ],
         "Resource":"*"
         ]
      }
   ]
}
```

Amazon Cloud Directory API Permissions: Actions, Resources, and Conditions Reference

When you are setting up Access Control (p. 176) and writing permissions policies that you can attach to an IAM identity (identity-based policies), you can use the following table as a reference. The list includes each Amazon Cloud Directory API operation, the corresponding actions for which you can grant permissions to perform the action, the AWS resource for which you can grant the permissions. You specify the actions in the policy's `Action` field and the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your Amazon Cloud Directory policies to express conditions. For a complete list of AWS-wide keys, see Available Keys in the *IAM User Guide*.

Note
To specify an action, use the `clouddirectory:` prefix followed by the API operation name (for example, `clouddirectory:CreateDirectory`).

AWS Directory Service API Permissions: Actions, Resources, and Conditions Reference

When you are setting up Access Control (p. 176) and writing permissions policies that you can attach to an IAM identity (identity-based policies), you can use the following table as a reference. The list includes each AWS Directory Service API operation, the corresponding actions for which you can grant permissions to perform the action, the AWS resource for which you can grant the permissions. You specify the actions in the policy's `Action` field and the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your AWS Directory Service policies to express conditions. For a complete list of AWS-wide keys, see Available Keys in the *IAM User Guide*.

Note
To specify an action, use the `ds:` prefix followed by the API operation name (for example, `ds:CreateDirectory`).

Related Topics

- Access Control (p. 176)

Troubleshooting AWS Directory Service Administration Issues

The following can help you troubleshoot some common issues you might encounter when creating or using your directory.

Simple AD

Here are some common problems with Simple AD.

Topics

I am not able to update the DNS name or IP address of an instance joined to my domain (DNS dynamic update)

DNS dynamic updates are not supported in Simple AD domains. You can instead make the changes directly by connecting to your directory using DNS Manager on an instance that is joined to your domain.

I cannot log onto SQL Server using a SQL Server account

You might receive an error if you attempt to use SQL Server Management Studio (SSMS) with a SQL Server account to log into SQL Server running on a Windows 2012 R2 EC2 instance or in Amazon RDS. The issue occurs when SSMS is run as a domain user and can result in the error "Login failed for user," even when valid credentials are provided. This is a known issue and AWS is actively working to resolve it.

To work around the issue, you can log into SQL Server with Windows Authentication instead of SQL Authentication. Or launch SSMS as a local user instead of a Simple AD domain user.

My directory is stuck in the "Requested" state

If you have a directory that has been in the "Requested" state for more than five minutes, try deleting the directory and recreating it. If this problem persists, contact the AWS Support Center.

I receive an "AZ Constrained" error when I create a directory

Some AWS accounts created before 2012 might have access to Availability Zones in the US East (N. Virginia), US West (N. California), or Asia Pacific (Tokyo) region that do not support AWS Directory Service directories. If you receive an error such as this when creating a directory, choose a subnet in a different Availability Zone and try to create the directory again.

Some of my users cannot authenticate with my directory

Your user accounts must have Kerberos preauthentication enabled. This is the default setting for new user accounts, and it should not be modified. For more information about this setting, go to Preauthentication on Microsoft TechNet.

AD Connector

Here are some common problems with AD Connector.

I receive a "DNS unavailable" error when I try to connect to my on-premises directory

You receive an error message similar to the following when connecting to your on-premises directory:

```
DNS unavailable (TCP port 53) for IP: <DNS IP address>
```

AD Connector must be able to communicate with your on-premises DNS servers via TCP and UDP over port 53. Verify that your security groups and on-premises firewalls allow TCP and UDP communication over this port. For more information, see AD Connector Prerequisites (p. 119).

I receive a "Connectivity issues detected" error when I try to connect to my on-premises directory

You receive an error message similar to the following when connecting to your on-premises directory:

```
Connectivity issues detected: LDAP unavailable (TCP port 389) for IP: <IP address>
Kerberos/authentication unavailable (TCP port 88) for IP: <IP address>
Please ensure that the listed ports are available and retry the operation.
```

AD Connector must be able to communicate with your on-premises domain controllers via TCP and UDP over the following ports. Verify that your security groups and on-premises firewalls allow TCP and UDP communication over these ports. For more information, see AD Connector Prerequisites (p. 119).

- 88 (Kerberos)
- 389 (LDAP)

I receive an "SRV record" error when I try to connect to my on-premises directory

You receive an error message similar to one or more of the following when connecting to your on-premises directory:

```
SRV record for LDAP does not exist for IP: <DNS IP address>

SRV record for Kerberos does not exist for IP: <DNS IP address>
```

AD Connector needs to obtain the `_ldap._tcp.<DnsDomainName>` and `_kerberos._tcp.<DnsDomainName>` SRV records when connecting to your directory. You will get this error if the service cannot obtain these records from the DNS servers that you specified when connecting to your directory. For more information about these SRV records, see SRV record requirements (p. 119).

My directory is stuck in the "Requested" state

If you have a directory that has been in the "Requested" state for more than five minutes, try deleting the directory and recreating it. If this problem persists, contact the AWS Support Center.

I receive an "AZ Constrained" error when I create a directory

Some AWS accounts created before 2012 might have access to Availability Zones in the US East (N. Virginia), US West (N. California), or Asia Pacific (Tokyo) region that do not support AWS Directory Service directories. If you receive an error such as this when creating a directory, choose a subnet in a different Availability Zone and try to create the directory again.

Some of my users cannot authenticate with my directory

Your user accounts must have Kerberos preauthentication enabled. This is the default setting for new user accounts, but it should not be modified. For more information about this setting, go to Preauthentication on Microsoft TechNet.

I receive an "Invalid Credentials" error when the service account used by AD Connector attempts to authenticate

This can occur if the hard drive on your domain controller runs out of space. Ensure that your domain controller's hard drives are not full.

Directory Status

The following are the various statuses for a directory. For more information, see Directory Status Reasons (p. 188).

Active

The directory is operating normally. No issues have been detected by the AWS Directory Service for your directory.

Creating

The directory is currently being created. Directory creation typically take 5 to 10 minutes but may vary depending on the system load.

Deleted

The directory has been deleted. All resources for the directory have been released. Once a directory enters this state, it cannot be recovered.

Deleting

The directory is currently being deleted. The directory will remain in this state until it has been completely deleted. Once a directory enters this state, the delete operation cannot be cancelled, and the directory cannot be recovered.

Failed

The directory could not be created. Please delete this directory. If this problem persists, please contact the AWS Support Center.

Impaired

The directory is running in a degraded state. One or more issues have been detected, and not all directory operations may be working at full operational capacity.

Inoperable

The directory is not functional. All directory endpoints have reported issues.

Requested

A request to create your directory is currently pending.

RestoreFailed

Restoring the directory from a snapshot failed. Please retry the restore operation. If this continues, try a different snapshot, or contact the AWS Support Center.

Restoring

The directory is currently being restored from an automatic or manual snapshot. Restoring from a snapshot typically takes several minutes, depending on the size of the directory data in the snapshot.

Directory Status Reasons

When a directory is impaired or inoperable, the directory status message contains additional information. The status message is displayed in the AWS Directory Service console, or returned in the `DirectoryDescription.StageReason` member by the `DescribeDirectories` API. For more information about the directory status, see Directory Status (p. 187).

The following are the status messages for a Simple AD directory:

Topics

The directory service's elastic network interface is not attached

Description

The critical elastic network interface (ENI) that was created on your behalf during directory creation to establish network connectivity with your VPC is not attached to the directory instance. AWS applications backed by this directory will not be functional. Your directory cannot connect to your on-premises network.

Troubleshooting

If the ENI is detached but still exists, contact AWS Support. If the ENI is deleted, there is no way to resolve the issue and your directory is permanently unusable. You must delete the directory and create a new one.

Issue(s) detected by instance

Description

An internal error was detected by the instance. This usually signifies that the monitoring service is actively attempting to recover the impaired instances.

Troubleshooting

In most cases, this is a transient issue, and the directory eventually returns to the Active state. If the problem persists, contact AWS Support for more assistance.

The critical AWS Directory Service reserved user is missing from the directory

Description

When a Simple AD is created, AWS Directory Service creates a service account in the directory with the name `AWSAdmin`*`D-xxxxxxxxxx`*. This error is received when this service account cannot be found. Without this account, AWS Directory Service cannot perform administrative functions on the directory, rendering the directory unusable.

Troubleshooting

To correct this issue, restore the directory to a previous snapshot that was created before the service account was deleted. Automatic snapshots are taken of your Simple AD directory one time a day. If it has been more than five days after this account was deleted, you may not be able to restore the directory to a state where this account exists. If you are not able to restore the directory from a snapshot where this account exists, your directory may become permanently unusable. If this is the case, you must delete your directory and create a new one.

The critical AWS Directory Service reserved user needs to belong to the Domain Admins AD group

Description

When a Simple AD is created, AWS Directory Service creates a service account in the directory with the name `AWSAdmin`*`D-xxxxxxxxxx`*. This error is received when this service account is not a member of the `Domain Admins` group. Membership in this group is needed to give AWS Directory Service the privileges it needs to perform maintenance and recovery operations, such as transferring FSMO roles, domain joining new directory controllers, and restoring from snapshots.

Troubleshooting

Use the Active Directory Users and Computers tool to re-add the service account to the `Domain Admins` group.

The critical AWS Directory Service reserved user is disabled

Description

When a Simple AD is created, AWS Directory Service creates a service account in the directory with the name `AWSAdmin`*`D-xxxxxxxxxx`*. This error is received when this service account is disabled. This account must be enabled so that AWS Directory Service can perform maintenance and recovery operations on the directory.

Troubleshooting

Use the Active Directory Users and Computers tool to re-enable the service account.

The main domain controller does not have all FSMO roles

Description

All the FSMO roles are not owned by the Simple AD directory controller. AWS Directory Service cannot guarantee certain behavior and functionality if the FSMO roles do not belong to the correct Simple AD directory controller.

Troubleshooting

Use Active Directory tools to move the FSMO roles back to the original working directory controller. For more information about moving the FSMO roles, go to https://support.microsoft.com/en-us/kb/324801. If this does not correct the problem, please contact AWS Support for more assistance.

Domain controller replication failures

Description

The Simple AD directory controllers are failing to replicate with one another. This can be caused by one or more of the following issues:

- The security groups for the directory controllers does not have the correct ports open.

- The network ACLs are too restrictive.
- The VPC route table is not routing network traffic between the directory controllers correctly.
- Another instance has been promoted to a domain controller in the directory.

Troubleshooting

For more information about your VPC network requirements, see either Microsoft AD Microsoft AD Prerequisites (p. 56), AD Connector AD Connector Prerequisites (p. 119), or Simple AD Simple AD Prerequisites (p. 133). If there is an unknown domain controller in your directory, you must demote it. If your VPC network setup is correct, but the error persists, please contact AWS Support for more assistance.

Schema Extension Errors

The following can help you troubleshoot some error messages you might encounter when extending the schema for your Microsoft AD directory.

Referral

Error

Add error on entry starting on line 1: Referral The server side error is: 0x202b A referral was returned from the server. The extended server error is: 0000202B: RefErr: DSID-0310082F, data 0, 1 access points \tref 1: 'example.com' Number of Objects Modified: 0

Troubleshooting

Ensure that all of the distinguished name fields have the correct domain name. In the example above, `DC=example,dc=com` should be replaced with the `DistinguishedName` shown by the cmdlet `Get-ADDomain`.

Unable to Read Import File

Error

Unable to read the import file. Number of Objects Modified: 0

Troubleshooting

The imported LDIF file is empty (0 bytes). Ensure the correct file was uploaded.

Syntax Error

Error

There is a syntax error in the input file Failed on line 21. The last token starts with 'q'. Number of Objects Modified: 0

Troubleshooting

The text on line 21 is not formatted correctly. The first letter of the invalid text is `A`. Update line 21 with valid LDIF syntax. For more information about how to format the LDIF file, see Step 1: Create Your LDIF File (p. 62).

Attribute or Value Exists

Error

Add error on entry starting on line 1: Attribute Or Value Exists The server side error is: 0x2083 The specified value already exists. The extended server error is: 00002083: AtrErr: DSID-03151830, #1: \t0: 00002083: DSID-03151830, problem 1006 (ATT_OR_VALUE_EXISTS), data 0, Att 20019 (mayContain):len 4 Number of Objects Modified: 0

Troubleshooting

The schema change has already been applied.

No Such Attribute

Error

Add error on entry starting on line 1: No Such Attribute The server side error is: 0x2085 The attribute value cannot be removed because it is not present on the object. The extended server error is: 00002085: AtrErr: DSID-03152367, #1: \t0: 00002085: DSID-03152367, problem 1001 (NO_ATTRIBUTE_OR_VAL), data 0, Att 20019 (mayContain):len 4 Number of Objects Modified: 0

Troubleshooting

The LDIF file is trying to remove an attribute from a class, but that attribute is currently not attached to the class. Schema change was probably already applied.

Error

Add error on entry starting on line 41: No Such Attribute 0x57 The parameter is incorrect. The extended server error is: 0x208d Directory object not found. The extended server error is: "00000057: LdapErr: DSID-0C090D8A, comment: Error in attribute conversion operation, data 0, v2580" Number of Objects Modified: 0

Troubleshooting

The attribute listed on line 41 is incorrect. Double-check the spelling.

No Such Object

Error

Add error on entry starting on line 1: No Such Object The server side error is: 0x208d Directory object not found. The extended server error is: 0000208D: NameErr: DSID-03100238, problem 2001 (NO_OBJECT), data 0, best match of: 'CN=Schema,CN=Configuration,DC=example,DC=com' Number of Objects Modified: 0

Troubleshooting

The object referenced by the distinguished name (DN) does not exist.

Trust Creation Status Reasons

When trust creation fails, the status message contains additional information. Here's some help understanding what those messages mean.

Access is denied

Access was denied when trying to create the trust. Either the trust password is incorrect or the remote domain’s security settings do not allow a trust to be configured. Ensure that you are using the same trust password that you used when creating the corresponding trust on the remote domain. Also confirm that your domain security settings allow for trust creation.

The specified domain name does not exist or could not be contacted

To solve this problem, ensure the security group settings for your domain and access control list (ACL) for your VPC are correct and you have accurately entered the information for your conditional forwarder. For more information about security requirements, see When to Create a Trust Relationship (p. 66).

If this does not solve the issue, it is possible that information for a previously created conditional forwarder has been cached, preventing the creation of a new trust. Please wait several minutes and then try creating the trust and conditional forwarder again.

General tool for testing trusts

The DirectoryServicePortTest testing tool can be helpful when troubleshooting trust creation issues between Microsoft AD and on-premises Active Directory. For an example on how the tool can be used, see Connect Verification (p. 123).

AWS Directory Service Limits

The following are the default limits for AWS Directory Service. Each limit is per region unless otherwise noted.

Amazon Cloud Directory

Schema and Directory Limits

Limit/Concept	Quantity
Number of attributes per facet (including required)	1000
Number of facets per object	5
Number of unique indexes an object is attached	3
Number of facets per schema	30
Number of rules per attribute	5
Number of attributes with default values per facet	10
Number of required attributes per facet	30
Number of development schemas	20
Number of published schemas	20
Number of applied schemas	5
Number of directories	100
Max page elements	30
Max input size (all inputs combined)	200 KB
Max response size (all outputs combined)	1 MB
Schema JSON file size limit	200 KB
Facet name length	64 UTF-8 encoded bytes
Directory name length	64 UTF-8 encoded bytes
Schema name length	64 UTF-8 encoded bytes

Object Limits

Limit/Concept	Quantity
Number of written objects	20 per API call

Limit/Concept	Quantity
Number of read objects	200 per API call
Number of written attribute values	1000 per API call
Number of read attribute values	1000 per API call
Path depth	15
Max objects in a single page	30
Max input size (all inputs combined)	200 KB
Max response size (all outputs combined)	1 MB
Policy size limit	10 KB
Edge or link name length	64 UTF-8 encoded bytes
Value length for indexed attributes	64 UTF-8 encoded bytes
Value length for non-indexed attributes	2KB
Number of policies attached to an object	4

Limits on batch operations

There are no limits on the number of operations you can call inside a batch. For more information, see Limits on Batch operations (p. 50).

Limits that cannot be modified

Amazon Cloud Directory limits that cannot be changed or increased, include:

- Facet name length
- Directory name length
- Schema name length
- Max page elements
- Edge or link name length
- Value length for indexed attributes

Simple AD

AWS Directory Service Limits

Resource	Default Limit
Simple AD directories	10
Manual snapshots	5 per Simple AD

AWS Directory Service for Microsoft Active Directory

AWS Directory Service Limits

Resource	Default Limit
Microsoft AD directories	10
Manual snapshots	5 per Microsoft AD

AD Connector

AWS Directory Service Limits

Resource	Default Limit
AD Connector directories	10

Increase Your Limit

Perform the following steps to increase your limit for a region.

To request a limit increase for a region

1. Go to the AWS Support Center page, sign in, if necessary, and click **Open a new case**.
2. Under **Regarding**, select **Service Limit Increase**.
3. Under **Limit Type**, select **AWS Directory Service**.
4. Fill in all of the necessary fields in the form and click the button at the bottom of the page for your desired method of contact.

Document History

The following table describes the important changes since the last release of the *AWS Directory Service Administrator Guide*.

- **Latest documentation update:** August 9, 2017

Change	Description	Date Changed
Facet-based indexing	Added facet-based index section. For more information, see Indexing and Search (p. 42).	August 9, 2017
Batches	Updated information about batches for Amazon Cloud Directory. For more information, see Batches (p. 48).	July 26, 2017
Compliance	Added information about HIPAA and PCI compliance. For more information, see Amazon Cloud Directory Compliance (p. 47) and Manage Microsoft AD Compliance (p. 115).	July 14, 2017
Fine-grained password policies	Added new password policies content. For more information, see Manage Fine-Grained Password Policies in Microsoft AD (p. 111).	July 5, 2017
Additional domain controllers	Added information for adding more domain controllers to your Microsoft AD. For more information, see Deploy Additional Domain Controllers (p. 114).	June 30, 2017
Tutorials	Added new tutorials for testing a Microsoft AD lab environment. For more information, see Microsoft AD Test Lab Tutorials (p. 93).	June 21, 2017
Typed links	Added new typed links content for Amazon Cloud Directory. For more information, see Objects and Links (p. 16).	May 31, 2017
MFA with Microsoft AD	Added documentation for using MFA with Microsoft AD. For more information, see Multi-Factor Authentication (p. 109).	February 13, 2017
Amazon Cloud Directory	New directory type introduced. For more information, see Amazon Cloud Directory (p. 11).	January 26, 2017
Schema Extensions	Added documentation for schema extensions with AWS Directory Service for Microsoft Active Directory. For more information, see Schema Extensions (p. 60).	November 14, 2016
Major reorganization of the Directory Service Admin Guide	Reorganized the content to more directly map to customer needs.	November 14, 2016
Authorization and Authentication	Added additional documentation for using IAM with Directory Services.	February 25, 2016

Change	Description	Date Changed
SNS notifications	Added documentation for SNS notifications. For more information, see Get Notified of Directory Status Updates Using Amazon SNS (p. 140).	February 25, 2016
Microsoft AD	Added documentation for Microsoft AD, and combined guides into a single guide.	November 17, 2015
Allow Linux instances to be joined to a Simple AD directory	Added documentation for joining a Linux instance to a Simple AD directory. For more information, see Manually Add a Linux Instance (Simple AD and Microsoft AD) (p. 158).	July 23, 2015
Guide separation	The *AWS Directory Service administration guide* is split into separate guides, the *Simple AD guide* and the *AD Connector guide*.	July 14, 2015
Single sign-on support	Added single sign-on documentation. For more information, see Single Sign-On (p. 168).	March 31, 2015
New guide	This is the first release of the *AWS Directory Service Administration Guide*.	October 21, 2014