Quest® On Demand Migration

Entra ID for Devices User Guide

© 2026 Quest Software Inc. ALL RIGHTS RESERVED.

This guide contains proprietary information protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Quest Software Inc.

The information in this document is provided in connection with Quest Software products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Quest Software products. EXCEPT AS SET FORTH IN THE TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, QUEST SOFTWARE ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL QUEST SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF QUEST SOFTWARE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Quest Software makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Quest Software does not make any commitment to update the information contained in this document.

If you have any questions regarding your potential use of this material, contact:

Quest Software Inc.

Attn: LEGAL Dept

10801 N. Mopac Expressway

Building 3, Suite 400

Austin, TX 78759

Refer to our Web site (https://www.quest.com) for regional and international office information.

Patents

Quest Software is proud of our advanced technology. Patents and pending patents may apply to this product. For the most current information about applicable patents for this product, please visit our website at https://www.quest.com/legal.

Trademarks

Quest, the Quest logo, and Join the Innovation are trademarks and registered trademarks of Quest Software Inc. For a complete list of Quest marks, visit https://www.quest.com/legal/trademark-information.aspx. All other trademarks and registered trademarks are property of their respective owners.

Legend

CAUTION: A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

IMPORTANT, **NOTE**, **TIP**, **MOBILE**, or **VIDEO**: An information icon indicates supporting information.

Quest On Demand Entra ID for Devices User Guide
Updated - May 2026

Contents

Introduction **4**
Planning the Migration Project 4
Requirements 5

Setup **8**
Workflows 8
Environments 9
Microsoft Entra Join Profiles 10
Migration Waves 11
Configurations 12
Downloads 12
Installing the Active Directory Agent 13
Repositories 18
Product Licensing 18

Managing Devices and Servers **19**

Frequently Asked Questions **23**

Additional Info **25**
Entra ID for Devices Architecture 25
Troubleshooting 27
Cutover Job Result Codes 28
Upload Logs Result Codes 28
CSV Import File Script 29

About us **46**
Technical support resources 46

1

Introduction

On Demand Migration Entra ID for Devices helps you transition from hybrid AD to the cloud by migrating Windows 10 and Windows 11 devices to Entra ID without reimaging and rebuilding profiles. This SaaS solution automatically creates workflows for your project using a simple step-by-step setup wizard, with no servers or complex configuration required.

This is one of three SaaS solutions Quest offers for device migration. Please reference the user guide for the solution you have purchased:

- On Demand Migration Active Directory User Guide: Directory Sync, coexistence, domain rewrite, domain move, AD migration, device migrations
- On Demand Migration Active Directory Express User Guide: Device migrations that do not require coexistence
- On Demand Migration Entra ID for Devices User Guide: Device migrations from on-prem and hybrid AD to Entra ID that do not include an AD migration

Planning the Migration Project

A typical migration project using Entra ID for Devices can be broken up into four phases.

- Phase 1: Create a Workflow with the Workflow Wizard
- Phase 2: ReACL Devices
- Phase 3: Cutover Devices
- Phase 4: Cleanup

This user guide walks you through the steps required to complete each phase, which can also be used to migrate devices from AD environments to Entra environments. The Microsoft Entra ID Device Join Quick Start Guide walks you through the process of configuring and performing migrations for AD to Entra migrations.

Best practices for each phase of the migration project are presented below:

Phase 1: Create a Workflow with the Workflow Wizard

- The wizard begins with selecting the environment type for both source and target environments. Only Local is available for the Source Environment Type and only Cloud is available for Target Environment Type.
- Workflow options are configured and CSV files containing the Users, Groups and Devices to import and the Users and Groups to match in the Target are uploaded.
- Microsoft Entra ID Join Profiles and Repositories are configured.

Phase 2: ReACL Devices

- Run a ReACL (file level re-permissioning) job on as many Devices as possible early in the process.
- ReACL is a non-destructive process that can be repeated as often as necessary up until Cutover in Phase 5.
- Troubleshoot any Devices with ReACL jobs which did not complete successfully.
- Run a ReACL job again close to the actual Cutover date. This will allow you to complete most of the ReACL process early and provide time to resolve any issues with things such as anti-virus software and Group Policies.

Phase 3: Cutover Devices

- Using some test Devices, Users, and Groups, verify a successful Device Cutover.
- Typically, a final ReACL job should be run the weekend before the scheduled Cutover to ensure any new Users and other changes are processed.
- A workstation reboot is required after the target account is enabled, the source account is disabled, and the Cutover is complete. This is usually completed in the evening when fewer end-users are impacted. Any impacted end-users should be alerted that this reboot is necessary.
- Optionally, use the Autopilot Cleanup option to prepare the AutoPilot-provisioned device for migation. This must be done before the cutover if the source Entra ID Joined device is Autopilot-provisioned and the Entra ID Join Profile has the Autopilot/Intune Cleanup option selected.

Phase 4: Cleanup

NOTE: The Cleanup process typically occurs several months after the completion of the project.

- The Cleanup phase typically takes place about two months after all Device Cutovers are complete. During the Cleanup phase, all permissions should be removed from the source domain and then the Active Directory agent should be removed from the Devices.
- Optionally, use the Set Intune Primary User action after the Device Cutover is completed.

Requirements

Networking

Outbound Internet Access

By default, each computer being migrated will require outbound access to the public Internet to securely communicate with the On Demand Migration services.

NOTE: If your organization requires computers communicate externally using a web proxy see our web proxy configuration requirements.

Application Ports

Each computer being migrated will require the Active Directory device agent and this agent will communicate to the On Demand Migration services, outbound over ports:

- 80
- 443
- 3030

Domain Controller Ports

Active Directory migrations also require a variety of Microsoft defined ports for communication between domain controllers. For a complete list of required ports, click here.

NOTE: For complete port information, review the Service overview and network port requirements for Windows documentation from Microsoft Support.

Accounts

Microsoft Entra ID Application Account

- When creating a new Cloud Environment, an account with the Global Administrator Role is required to grant permissions and establish a connection.

Devices

The following is required for any Active Directory Computer(s) (devices) that will be migrated.

Device Agents

Each Active Directory Computer that will be migrated must have an agent installed on the workstation to orchestrate local jobs that must occur to prepare and execute the workstation's domain move.

Operating Systems

All computers or servers being migrated to the new domain must run one of the following operating systems:

- Windows 10
- Windows 11

NOTE: Entra ID Device Join is only supported for Windows 10 and 11. Any operating system running on the ARM Processor will not be supported.

PowerShell

- All client operating systems must have at least PowerShell 2.0 installed.

.NET Framework

- All Devices must have .NET Framework 4.7.2 or newer installed. This will appear as ".NET 4.7.2 Extended" in the add/remove programs list.
- If not present, an appropriate version of .NET Framework will be installed during agent installation if an internet connection is available.

Web Proxy

Some organizations may require all computers communicating externally direct their traffic through a web proxy to centralize communications. Active Directory agents can be configured to use a web proxy for communication to the

On Demand Migration cloud services.

Proxy Server

- At least one (1) standard web proxy that supports http/TCP traffic.

Proxy Address

- The associated web proxy URL must be defined during configuration of the device agent.

Security

- If accessing the web proxy requires an additional username and password this will be required during configuration of the device agent.

Ports

All agents configured to use a web proxy will utilize the following outbound TCP ports:

- 80
- 443

NOTE: Agents configured to use a web proxy will not require UDP port 3030. For more information, see the Web Proxy Configuration under Architecture.

Additional bandwidth overhead may occur when a web proxy is utilized to centralize all traffic.

2

Setup

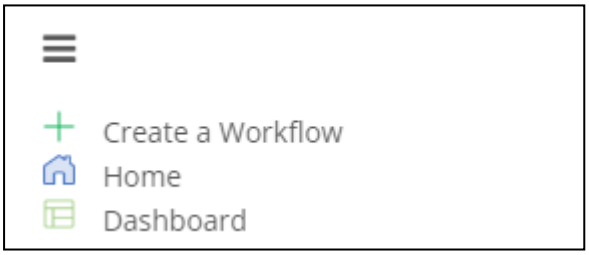
Workflows

A workflow is a configurable series of steps that provides an easy automation framework to connect and manage Directory object synchronization. Activities such as creating, updating and deleting objects along with property/attribute synchronization and transformation.

How do I create and manage Workflows?

To create a Worklfow, simply open the left navigation menu and click **Create a Workflow** in the side navigation menu, see *figure 1*, or click the **New** button under Workflows on the dashboard. The Workflow Wizard will open and will guide you through the creation of the Workflow.

To manage a Workflow, click the **Manage** button under Workflows on the dashboard.



What should be entered as the Workflow Name?

You can name your workflow anything you'd like but remember that you may be referencing the same environment in multiple workflows. We suggest a name that generally describes the flow of objects. Then use the description field for the distinguishing characteristics. After this step, the wizard will guide you through all the necessary components that will make up your workflow.

What are the steps to create a Workflow?

Entra ID for Devices uses a wizard interface to guide you through the steps of creating and configuring the Workflow. To launch the wizard, click **Create a Workflow** in the side navigation menu or click the **New** button under Workflows on the dashboard.

Steps of the Workflow Wizard:

1. Select Environment Types - You are prompted to select the environment type for both source and target environments. Options are Local (a traditional on-premises Active Directory environment.) and Cloud (a Microsoft Entra ID environment.) See the Environments topic for more information.

2. Configure Source Environment:
    a. Provide source environment name - Provide a descriptive name for the source environment that makes it easy to identify. For example, the Bluefish Resort on-premises Active Directory environment could be named bluefishresort.com.
3. Configure Target Environment:
    a. Provide target environment name - Provide a descriptive name for the target environment that makes it easy to identify. For example, the Bluefish Resort on-premises Active Directory environment could be named bluefishresort.com.
    b. Connect to your Cloud Environment - Add a commercial tenant. Important: A Service Principal will be created in the tenant.
4. Configure Workflow:
    a. Name the Workflow - You can name your workflow anything you'd like but remember that you may be referencing the same environment in multiple workflows. We suggest a name that generally describes the flow of objects. Then use the description field for the distinguishing characteristics.
    b. Users File to import for scoping and matching - Upload a CSV containing the Users to import and the Users to match in the Target. Note that Import file must include ObjectID's. The file must include at least one user to continue.
    c. Groups File to import for scoping and matching - Upload a CSV containing the Groups to import and the Groups to match in the Target. Note that Import file must include ObjectID's
    d. Devices File to import for scoping and matching - Upload a CSV containing the Devices to import. Note that Import file must include ObjectID's
5. Microsoft Entra ID Join Profile - Microsoft Entra ID Join Provisioning Package file contains the target Microsoft Entra ID information used during Microsoft Entra ID Device cutover process.
6. Repositories - Repositories are specified storage locations on your network used for the following specific job types. See the Repositories topic for more information.
7. Downloads - See the Downloads topic for more information.
8. Summary - Please verify that all of the information has been correctly entered. Click the **Edit** button next to information that needs to be changed. Click **Finish**.

Environments

If a workflow is a series of action steps, an environment is the receiver of those actions. In the Workflow Wizard, you will choose two environments, a source and a target, that the workflow will take actions against.

Where do I manage Environments?

To manage environments, select an environment from the Environment Summary table on the dashboard and click the **Manage** button.

How are Environments added?

The Workflow Wizard will guide you in adding a Source and Target Environment.

How do you export a list of Users, Groups, Contacts, and Devices in an environment?

On the dashboard, click the **Manage** button under the Environment Summary table. On the Environments page, select an environment and click the **Details** button. Expand an object list and click the **Export** button to download a CSV file of the Users, Groups, Contacts, and Devices.

How do you unmatch Users, Groups, Contacts, and Devices so they will not be synchronized?

On the dashboard, click the **Manage** button under the Environment Summary table. On the Environments page, select an environment and click the **Details** button. Expand an object list.select an object in the table and click the **Unmatch** button. The Match Status for the object will change to "Unmatched" and the object will not be synchronized.

Microsoft Entra Join Profiles

A Microsoft Entra Join Profile is a collection of settings used to manage the Entra join process during Device Cutover which can be defined once and then applied to multiple Devices. Microsoft Entra Join Profiles are used for AD to Entra device migrations.

How are Microsoft Entra Join Profiles created?

To add an Microsoft Entra Join Profile:

1. On the *Microsoft Entra Join* section of the *Profiles* page, Click the **Add** button. The *Add Your Microsoft Entra Join Profile* window appears.
2. Enter a **Profile Name** to identify this Microsoft Entra Join Profile.
3. Enter a value in the following field:
    - **Bulk Enrollment Package File Name** - The name of the Microsoft Entra bulk enrollment package in packagename.ppkg format, which has been created by the client administrator using the Windows Configuration Designer and copied to the network share defined in the Microsoft Entra Bulk Enrollment Repository
4. Select an option from the following drop-down list:
    - **Target Environment** – The cloud-only Azure environment associated with the Azure bulk enrollment package used in this Profile
5. Select a **Device Name Option**:
    - If you choose **Device Name Defined Per Provisioning Package**, the device will be migrated to Entra using the dynamic naming convention configured in the Microsoft Entra bulk enrollment package used in this Profile
    - If you choose **Keep Original Device Name**, the dynamic name assigned by the Microsoft Entra bulk enrollment package will be overwritten and replaced with the original device name when migrating to Entra
6. Select the **Enroll Into Intune Management** option to enroll the device for Intune management with the first logged on user after cutover as the PrimaryUser.

7. Select the **Auto-Pilot/Intune Cleanup** option to clear existing Auto-Pilot/Intune provisioning information from the device as part of the cutover.
8. Select the **Source Directory is Active Directory Joined or Hybrid Microsoft Entra ID Joined** option if you wish to include Active Directory Joined or Hybrid Microsoft Entra Joined devices.
9. Enter values in the following fields under Source Domain Credentials:
    - **FQDN of Domain** - The domain FQDN of the source in source.domain.com format.
    - **Username** - The username to access the source domain in domain\username or UPN (username@domain.com) format.
    - **Password** - The password credential to access the source domain.
10. Under *Preflight Check Validation*, select the **Skip Source Local Active Directory Validation** option to not validate the source local Active Directory.
11. Click **Save Profile**. The Microsoft Entra Join Profile is added to the list.

Migration Waves

A Migration Wave in Active Directory is a named logical grouping of Devices. This can be a useful tool for organizing, tracking, and staging your migrations.

How do I manage Migration Waves?

There are two ways to manage Migration Waves in Active Directory. First, you can add Devices to a Wave by applying the ‘Set to Migration Wave’ Action to Devices from the Ready Devices table. The other way to manager Migration Waves is from the Migration Waves page which is accessible through ‘Waves’ in the left navigation menu. On that page you can create new Waves, Edit Wave names, Remove empty waves, and view how many Devices are in a Wave.

What can I do with a Migration Wave?

Migrations Waves in Active Directory can be used to filter the Ready Devices table to view Device status and apply Actions.

How do I create a New Migration Wave?

1. Login to Active Directory.
2. Select "Waves" in the left navigation menu.
3. A new page will open listing your current migration waves.
4. Click the Add icon.
5. Name the Migration Wave, remember use a logical name representing the migration event.
6. Click “Save.”
7. Now that the Migration Wave is created, Devices can be added to the Wave by applying the ‘Set to Migration Wave’ Action to Devices from the Ready Devices table.

How do I remove a Migration Wave?

1. Login to Active Directory.
2. Select "Waves" in the left navigation menu.
3. A new page will open listing your current migration waves.
4. Select one or more wave in the table.
5. Click the Remove icon.
6. Click "Yes" to confirm the removal.

How do I edit the name of a Migration Wave?

1. Login to Active Directory.
2. Select "Waves" in the left navigation menu.
3. A new page will open listing your current migration waves.
4. Select a wave in the table.
5. Click the Edit icon.
6. Edit the name of the Migration Wave.
7. Click “Save.”

How do I filter Devices by Migration Wave?

1. Login to Active Directory.
2. Select "Devices + Servers" in the left navigation menu.
3. On the Ready Devices tab, click the Filter icon.
4. Select one or more wave under the Waves filter category.

Configurations

Downloads

To download a device agent:

1. Select an available agent version from the drop-down menu.
2. Click the **Download** button.
3. Use the browser options to save the agent installer package.

What are the Device Agent Service URL and Auth key used for?

The Device Agent Service URL and Auth Key as defined on the Downloads section of the Configurations page are provided to the Device Agents at install and allow them to connect to the correct customer’s On Demand Migration project. They are unique to the agents in a given client and all agents of the same client should use the same values. If installing the agent from the command line without UI the arguments for providing the Service URL and Auth Key are their names in all uppercase i.e. SERVICEURL and AUTHKEY respectively.

How are device agents automatically upgraded?

To automatically upgrade the device agents:

1. Click the **Enable** button at the bottom of the Device Agent section.

Installing the Active Directory Agent

Each Active Directory Computer (device) that will be migrated must have an agent installed on the workstation to orchestrate local jobs that must occur to prepare and execute the workstation's domain move.

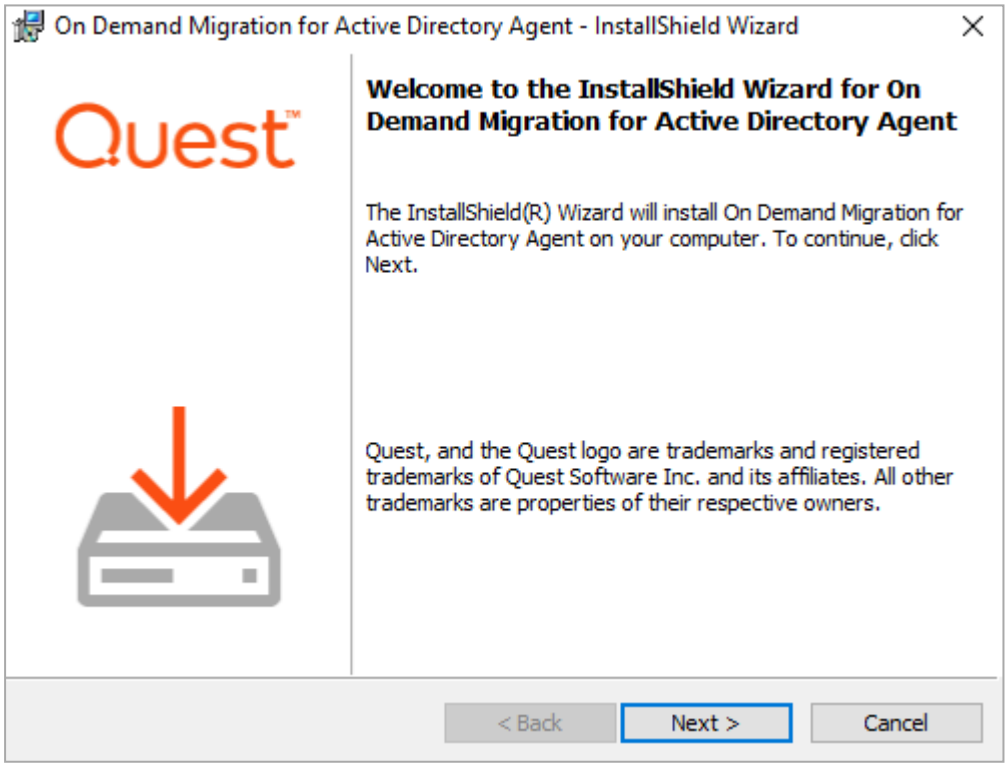
Refer to the Requirements for to verify all devices meet the requirements for agent installation.

The agent is available as an MSI package from the Downloads section of the Configurations page. You will also need the values of the Service URL and Auth Key found on that page.

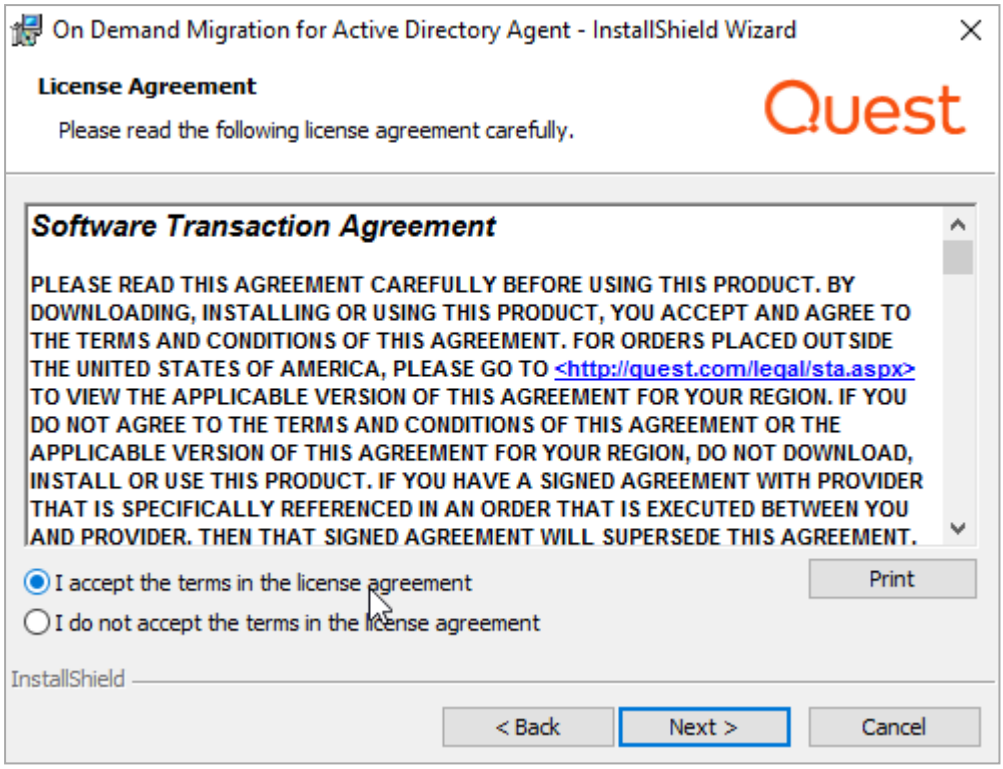
You can install the agent by running the MSI manually on the device, with a PowerShell command, or in bulk by using a GPO or other third-party delivery method.

How do you manually install the Active Directory Agent?

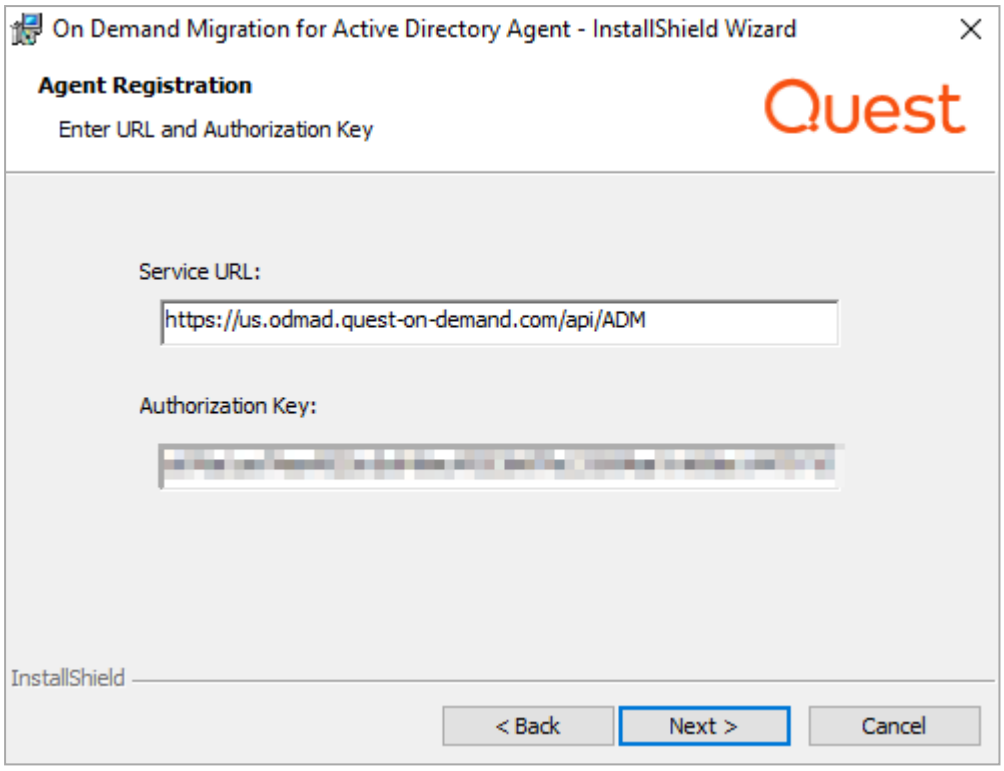
1. Download the Active Directory MSI file from the Downloads page.
2. Copy the Active Directory MSI file to each computer.
3. Double-click the file to open the installer.
4. On the Welcome screen, click **Next**.



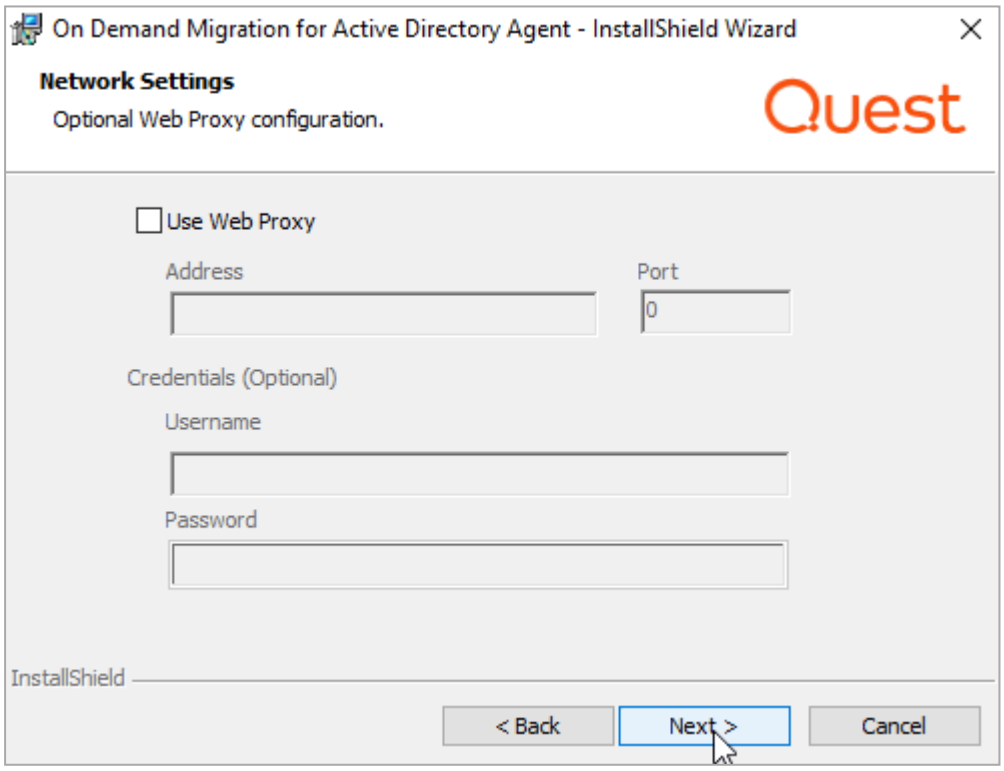
5. On the **License Agreement** screen, accept the agreement and click **Next**.



6. On the **Agent Registration** screen, enter the **Service URL** and **Authorization Key**, found on the Downloads page, and then click **Next**.



7. On the **Network Settings** screen, if using a Web Proxy, check **Use Web Proxy** and enter the Web Proxy settings. Click **Next**.



8. On the **Ready to Install the Program** screen, click **Install**.
9. When the install completes, click **Finish**.

> **NOTE:** Once the agent is installed and the service is running it will connect to the server within four hours. This delay is randomized and uniformly distributed to avoid overloading the server when large numbers of agents come online at the same time.

How do you install the Active Directory Agent using a PowerShell Command?

1. Download the Active Directory MSI file from the Downloads page. The Service URL and Auth Key values also found on the Downloads page are required.
2. Create and run the PowerShell command with the required SERVICEURL (Service URL) and AUTHKEY (Auth Key) values.

```
msiexec.exe /I 'C:\workspace\AD.Agent-20.3.1.1401.msi'
SERVICEURL=https://us.odmad.quest-on-demand.com/api/ADM
AUTHKEY=
######################################################################
```

3. Walk through the install wizard, filling out the needed information and click Finish when completed. The settings for using a customer web proxy for communications are optional.

As needed the installer can also be invoked in quiet mode with the /QN switch (requires running PowerShell as admin).

Additionally, it is possible to configure the agent to use a Web Proxy using the below command line arguments:

- **WEBPROXYENABLE** – Is a Web Proxy used? Values: Yes=1, No=0
- **WEBPROXYURL** – The Web Proxy Address
- **WEBPROXYPORT** – The Web Proxy Port
- **WEBPROXYUSER** – The optional Web Proxy Credentials Username
- **WEBPROXYPASS** – The optional Web Proxy Credentials Password

How do you install the Active Directory Agent using a GPO (Group Policy Object)?

1. To install the agent using a GPO you must convert the MSI package and the parameters into an MST file. One method to do this is using Microsoft Orca. Install Orca (available in Windows SDK Components for Windows Installer Developers). Orca will be used to create the necessary MST file.
2. Download the Active Directory Agent MSI file from the Downloads page.
3. Right-click on the MSI file and select **Edit with Orca**.
4. Once you have Orca opened, click on the **Transform** menu and select **New Transform**.
5. Next, navigate to the **Property** table and add the following:
   - Add a Row with property of **SERVICEURL** and the Service URL value found on the Downloads page.
   - Add a Row with property of **AUTHKEY** and the Auth Key value found on the Downloads page.
   - Optionally, the following properties and values can also be added to configure the agent to use a Web Proxy:

- **WEBPROXYENABLE** – Is a Web Proxy used? Values: Yes=1, No=0
- **WEBPROXYURL** – The Web Proxy Address
- **WEBPROXYPORT** – The Web Proxy Port
- **WEBPROXYUSER** – The optional Web Proxy Credentials Username
- **WEBPROXYPASS** – The optional Web Proxy Credentials Password

5. Click the **Transform** menu and select **Generate Transform** to complete the MST file creation. This MST file will be used in a later step.
6. Right-click on the Active Directory Agent MSI, point to **Share with**, and click on **specific people**.
7. Add a security group. The "authenticated users" group already includes all computers and is a good group to use. The group you add must have the shared Read permission and NTFS permission.
8. Click **Share**.
9. Click **Done**.
10. From the **Start** menu, point to **Administrative Tools** and click on **Group Policy Management**.
11. Right-click on the domain or OU you will be migrating and click on **Create a GPO in this domain, and link it here**.
12. In the New GPO dialog box, enter a **Name** for the GPO and click **OK**.
13. Click on the new GPO and click **OK**.
14. Right-click on the GPO and select **Edit**.
15. Open **Computer Configuration** > **Policies** > **Software Settings** and right-click on **Software Installation** and then point to **New** and click on **Package**.
16. In the **File Name** field, enter the UNC path to the MSI file and click **Open**.
17. Select the Active Directory Agent MSI file and click **Open**.
18. In the **Deploy Software** window, select the **Advanced** deployment method and click **OK**.
19. Under the **Modifications** tab, add the MST file you created earlier and click **OK**.

NOTE: The computer must be rebooted for the applied group policy to complete the agent installation.

How do you verify the GPO?

1. Log on to a workstation within the scope of the GPO using administrator credentials.
2. From a command prompt on the workstation, run **gpresult -r**
3. The Computer Settings section will display the applied group policy.

NOTE: A newly applied group policy will not immediately be displayed.

The Computer Settings section displays the applied group policy, but the agent installation is not completed until the computer is rebooted.

NOTE: If using the agent Auto-Upgrade feature and deployment software that uses MSI ProductCode based detection, the Auto-upgrade feature should be disabled after initial deployment or the detection method should verify via a folder path.

Repositories

Repositories are storage locations (network shares) which you configure on your network used for four specific job types: Agent Logs, Custom Downloads, Offline Domain Join, and Microsoft Entra Device Join. These job types access or create files in the defined locations. If you are not using these job types you do not need to configure Repositories.

How do I manage Repositories?

Repositories are managed from the Repositories section of the Configurations page. There you can view and copy the currently defined share path for each job type. You can update and save the configuration. Make sure that the defined network share path is routable from the devices being migrated.

What can I do with a Repository?

Repositories are used to locate files for four job types:

- Agent Logs – Agent logs are maintained on individual workstations. To centralize them so that review does not require logging into each machine, the Upload Logs Action can be applied to the Devices of interest. This will create a copy of the Device's logs in the defined network share.
- Custom Downloads – Custom Actions with a Download File Task can be used to download a specified file from the defined network share to the workstation. A common use of this job type is distributing a new VPN client to workstations after Cutover.
- A Custom Download Repository should not be used with the Offline Domain Join Action type when configuring custom actions.
- Microsoft Entra Bulk Enrollment – The Microsoft Entra ID Cutover Action relies on access to an Azure bulk enrollment package which has been created by a client administrator using the Windows Configuration Designer as described in the Microsoft Entra Device Join Quick Start Guide.

Product Licensing

The product licensing is based on the number of **unique source computer objects** per configured **environment workflow**. The licenses are consumed when a device migration related activity (ReACL Process or Cutover) is performed for each computer object.

3

Managing Devices and Servers

What is the Devices + Servers page used for?

The Servers + Devices screen allows the administrator to register devices and servers, set the ReACL profile, upload migration logs, and manage the device Discovery, ReACL, Cutover, and Cleanup processes.

Select the gear icon above the "Ready Devices" list to select the columns to display for the current session.

What is the difference between a "Ready Device" and a "Not Ready Device?"

"Ready Devices" are devices that have the necessary agent installed, are communicating, and are ready for actions to be scheduled. "Not Ready Devices" are devices which that have not yet had an agent installed and communicating.

NOTE: Initial agent registration is uniformly distributed over a four hour interval from agent start time.

What actions can be performed on Devices and Servers?

The following actions can be performed on devices by selecting the devices in the list, selecting an action from the drop-down list, and then clicking the **Apply Action** button.

- **Discovery**
  The Discovery process gathers properties (OS versions, network properties, and so on) from the device to allow additional future functionality. The first discovery process begins for a device when the device becomes registered with On Demand Migration which will automatically occur after the Device Agent has been installed, as long as the environment is properly configured.

  To select when the process will begin check **Do not start before** and then enter or select a date and time. If using the **Do not start before** option, the Discovery Status will be displayed as Queued in the Devices table and the "Do Not Start Before" column in the Device Jobs table will be populated with the selected date.
- **Set Target Environment**
  The Set Target Environment action provides the ability to specify the target environment for the selected devices. The ReACL, Cache Credential, Offline Domain Join, Cutover, Cleanup, Rollback, and ReACL Rollback actions cannot be started for the selected devices if the target environment is not set.

  To set to the target environment, select a target environment for the selected devices from the list and click **Save**. The target environment can be cleared by selecting **None** from the list.
- **ReACL**
  The ReACL process updates the Device's domain user profiles for use by the matching target user after cutover.

NOTE: It is recommended to remove or disable anti-virus software immediately prior to the ReACL process and only after a recent clean scan has been completed.

At least one group must be migrated to populate the map.gg file or the ReACL process will fail.

Before ReACL can occur, the target Users and Groups which have permissions set on the Device must be migrated to the target.

To select when the process will begin check **Do not start before** and then enter or select a date and time. If using the **Do not start before** option, the ReACL Status will be displayed as Queued in the Devices table and the "Do Not Start Before" column in the Device Jobs table will be populated with the selected date.

NOTE: Two checks are performed at the start of the ReACL process. The first check is for invalid Source Profiles, which will be logged as a WARNING and those profiles will be skipped. The second check is for invalid Target Profiles, where a user may have created a profile with the target account before their machine is ReACL'd and cutover. By default, this is logged as a FATAL ERROR and will halt the ReACL process. However, it can be changed to a WARNING with the –t switch passed by editing the command in SQL.

The ReACL Agent will automatically create two files on the device being ReACL'd, map.usr and map.gg. These files are used to find the source permissions and add the appropriate target permissions during the ReACL process. System groups, such as Domain\Domain Admins and Domain\Domain Users are included in the map.gg file for updating the group permissions during the ReACL process. If the Active Directory environment is non-English, the values in the sAMAccountName column of the BT_SystemGroup table in the SQL database will need to be changed after Directory Sync is installed to have the appropriate non-English values.

If the Mapped Network Drive is being mapped via GPO or using an integrated credential such as the current Windows logon session, ReACL will create a warning entry in the log "…WARNING: The UserName value for drive U was empty and could not be mapped to the target user." This warning does not mean that the mapped drive cannot be accessed after Cutover.

NOTE: The user profile ReACL process is decoupled from actions against files and folders.

ReACL will update all files and folders entries found on the machine except for the user profile folders, ntuser.dat, and usrclass.dat even if the user profiles option is selected in the ReACL profile.

Remaining ReACL activities against user profiles and registry are processed by a separate ReACL task when a cutover operation is performed.

- **Cache Credential**
  The Cache Credentials process assigns a Cache Credentials job to workstation(s). See the Credential Cache and Offline Domain Join topic for more information.
- **Offline Domain Join**
  The Offline Domain Join process is similar to the Cutover process for machines that are directly connected to the network. See the Credential Cache and Offline Domain Join topic for more information.

  CAUTION: Do not perform the Cutover process on Offline Domain Join workstations. The Offline Domain Join process takes the place of Cutover for workstations connecting via VPN.
- **Cutover**
  The Cutover process moves a Device from the source domain to the new target domain.

  Check **Ignore ReACL Status** to cutover the device regardless of the ReACL status (otherwise the cutover process will not proceed if there is an error with ReACL process).

Check **Do not start before** and then enter or select a date and time when the process will begin. If using the **Do not start before** option, the Cutover Status will be displayed as Queued in the devices table and the "Do Not Start Before" column in the Device Jobs table will be populated with the selected date. The Cutover process will begin as soon as possible if not using this option.

> **NOTE:** Devices should not be ReACL'd once they have been cutover to the Target. This is not a best practice and is not supported as this can cause problems with the registry and user profiles. Certificates are not migrated with Device Cutover.

- **Microsoft Entra ID Cutover**
  The Microsoft Entra ID Cutover process moves a Device from the source domain to an Azure target environment.

  Check **Ignore ReACL Status** to cutover the device regardless of the ReACL status (otherwise the cutover process will not proceed if there is an error with ReACL process).

  Check **Do not start before** and then enter or select a date and time when the process will begin. If using the **Do not start before** option, the Cutover Status will be displayed as Queued in the devices table and the "Do Not Start Before" column in the Device Jobs table will be populated with the selected date. The Cutover process will begin as soon as possible if not using this option.

  > **NOTE:** Devices should not be ReACL'd once they have been cutover to the target. This is not a best practice and is not supported as this can cause problems with the registry and user profiles.
  >
  > If the Entra ID Join profile has the **Auto-Pilot/Intune Cleanup** option selected, the Autopilot remove status must be completed before the Microsoft Entra ID Cutover action can be used.

- **AutoPilot Cleanup**
  The AutoPilot Cleanup action clears existing Auto-Pilot/Intune provisioning information from the device. This must be done before the cutover if the source Entra ID Joined device is Autopilot-provisioned and the Entra ID Join Profile has the Autopilot/Intune Cleanup option selected.

- **Set Intune Primary User**
  The Set Intune Primary User action is used after the Device Cutover is completed to set the primary user.

- **Cleanup**
  The Cleanup process removes the Source SIDs after the Cutover process completes.

  > **NOTE:** Cleanup should be done when the migration project is completed. Before running the Cleanup process if a trust is in place, the trust can be broken to test if any application permissions are broken.

  In the Job Options window, click **Apply** to begin the Cleanup process as soon as possible. To select when the process will begin check **Do not start before** and then enter or select a date and time. If using the **Do not start before** option, the Cleanup Status will be displayed as Queued in the Devices table and the "Do Not Start Before" column in the Device Jobs table will be populated with the selected date.

- **Upload Logs**
  Log files from the Active Directory Agent can be uploaded to the Active Directory Web Server using Microsoft BITS. To enable this functionality, the installer enables BITS Server Extensions for IIS and create a virtual directory called **DeviceLogs** where all uploaded files will be stored.

  In the Job Options window, click **Apply** to begin the Upload Logs process as soon as possible. To select when the process will begin check **Do not start before** and then enter or select a date and time. If using the **Do not start before** option, the Do Not Start Before column in the Device Jobs table will be populated with the selected date.

  The logs will be stored in the configured Agent Logs Repository

The device logs will be zipped, and the file names will be in the following format with a unique file name: SMART-WIN7X86-1_201573111235.zip

- **Rollback**
  The Rollback process moves a Device back to the original source domain and restores any modified network settings. The Device must have attempted Cutover for this explicit Rollback process to work.

  In the Job Options window, click **Apply** to begin the Rollback process as soon as possible. To select when the process will begin check **Do not start before** and then enter or select a date and time. If using the **Do not start before** option, the "Do Not Start Before" column in the Device Jobs table will be populated with the selected date.

  NOTE: Rollback is not supported for Entra ID Device Cutover.

- **ReACL Rollback**
  The ReACL Rollback process rolls back all changes made by the ReACL process. ReACL Rollback can be performed on Devices that have completed the ReACL process.

  In the Job Options window, click **Apply** to begin the ReACL Rollback process as soon as possible. To select when the process will begin check **Do not start before** and then enter or select a date and time when the process will begin. If using the **Do not start before** option, the "Do Not Start Before" column in the Device Jobs table will be populated with the selected date.

- **Status Resets**
  Use the Status Resets action to reset the statuses of the selected devices.

- **Set Device ReACL Profile**
  Use the Set Device ReACL Profile action to assign a Device ReACL profile to the selected devices.

- **Set Migration Wave**
  Use the Set Migration Wave action to set a migration wave to the selected devices.

- **View Jobs**
  Use the View Jobs action to view the device jobs of the selected devices.

  Note: Jobs can be canceled when the Status or Rollback Status is either Queued, Scheduled, Started, or In Progress.

- **View Profiles**
  Use the View Profiles action to view the profiles of the selected devices.

- **View Properties**
  After the Discovery process has been completed for a Device, you can view the properties of that Device.

  Click the **Export All** button to export the content of the window in Excel, text, CSV, or HTML format.

4

Frequently Asked Questions

What ports does the agent use to connect?

The Active Directory Agent uses three outbound ports

- TCP 443/80
- UDP 3030

I've installed the agent and the device isn't ready, what do I do?

On startup the Active Directory agent will phone home sometime in the first four hours. This communication offset time is an agent specific random and evenly distributed offset so the initial communications of a large number of devices set up at once will be spaced out and not overload the servers. If your device has the agent installed but you haven't waited up to four hours yet, wait up to four hours and then re-check the Ready Devices list to see if your device shows up before proceeding to other network troubleshooting operations.

How do I adjust the agent polling interval?

In Entra ID for Devices, the agent polling interval is not end user adjustable. The polling interval is every two minutes over UDP and every four hours over TCP. Should this amount of network traffic be an issue contact Support to investigate reducing the polling interval. Note that reducing the polling interval will cause queued migration actions to take longer to be picked up by the agents.

How many migration actions can I queue at once?

In Entra ID for Devices you can queue as many migration actions at once as you would like. There is a hard limit of 600 agents per each two minutes who will be notified by the job availability cache that they have a job waiting for them. Therefore the minimum amount of time which could be required for those queued migration actions to be picked up will be the number of queued migration actions divided by 600 and multiplied by two minutes.

Agent last contact time isn't updating every two minutes is something wrong?

In Entra ID for Devices, the Agent last contact time is updated based on the TCP connection with the back end. This will occur if a job is retrieved or every 4 hours as a fall back. It is not expected that the agent last contact time will update every two minutes even if the UDP requests to the job availability cache are functioning correctly.

The machine name was changed during the migration project, will it keep working?

Changing the machine name during a migration project may have unexpected implications. We recommend that you avoid doing so if possible. If not, you will need to delete the device from Entra ID for Devices and reinstall the agent. Use the Status Resets action to reset the registration status of the device, then set the Workflow to reconcile

on next run and run it. This will delete the old device from the database. Finally, reinstall the Active Directory Agent on that device and run the Workflow again to read it in.

How do I remove Devices from Entra ID for Devices?

To delete registered Devices from Entra ID for Devices, first use the Status Resets action to clear their registration status, then run the related Workflow with the setting 'Reconcile on next run' enabled. Any unregistered device records in scope of the Workflow will be removed from the database during a reconcile.

Can I migrate to and from GCC/GCCH tenants?

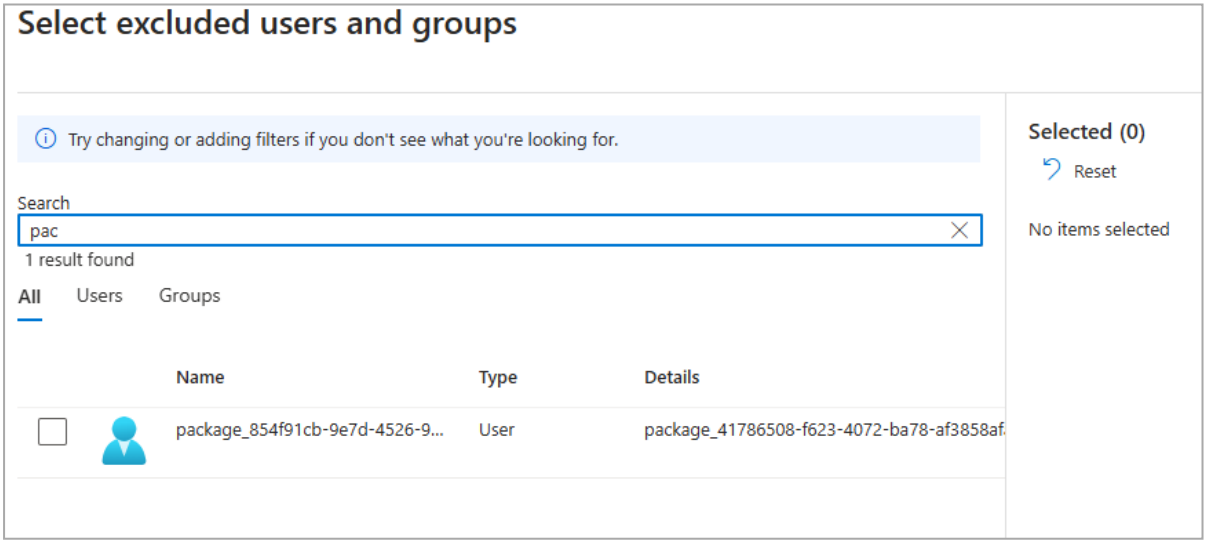
Active Directory supports GCC and GCCH as target environments, in addition to Commercial tenants. Only Commercial and GCC are supported as source environments. To enable GCC/GCCH Device Migration, please contact Quest Support for additional configuration described in the following article: Update GCC/GCCH tenant object's RID value for ReACL. This configuration is required for successfully switching the user profile during cutover.

Why isn't my device joining the target Entra ID, and why is it being removed automatically after a successful initial join?

If you have enabled automatic enrollment for Windows devices, you may need to create an exclusion for the provisioning package account. This exclusion is necessary when auto-enrollment is active.

With these settings enabled, the device will be automatically joined when the provisioning package is installed. However, it may also be removed immediately if the computer or account is not compliant with policy requirements. This commonly occurs when Multi-Factor Authentication (MFA) is enforced, as the provisioning package account is typically not MFA-compliant.

To resolve this, you'll need to create an MFA exclusion. Check the target MFA policy and add an exclusion for the provisioning package. When adding the exclusion, search for "package ...", you should see a result similar to the provisioning package account.



Add the provisioning package to the exclusion list, wait approximately 10 minutes for the changes to replicate, and then try provisioning the device again. Finally, review the Audit Logs to see if the behavior has changed.

5

Additional Info

Entra ID for Devices Architecture

The first step towards success on a project using Entra ID for Devices is to understand the product architecture and how this architecture will operate in your environment.

Entra ID for Devices consists of the following components:

- A directory synchronization engine
- A REST based web service
- A management interface
- A lightweight agent for workstations and member servers

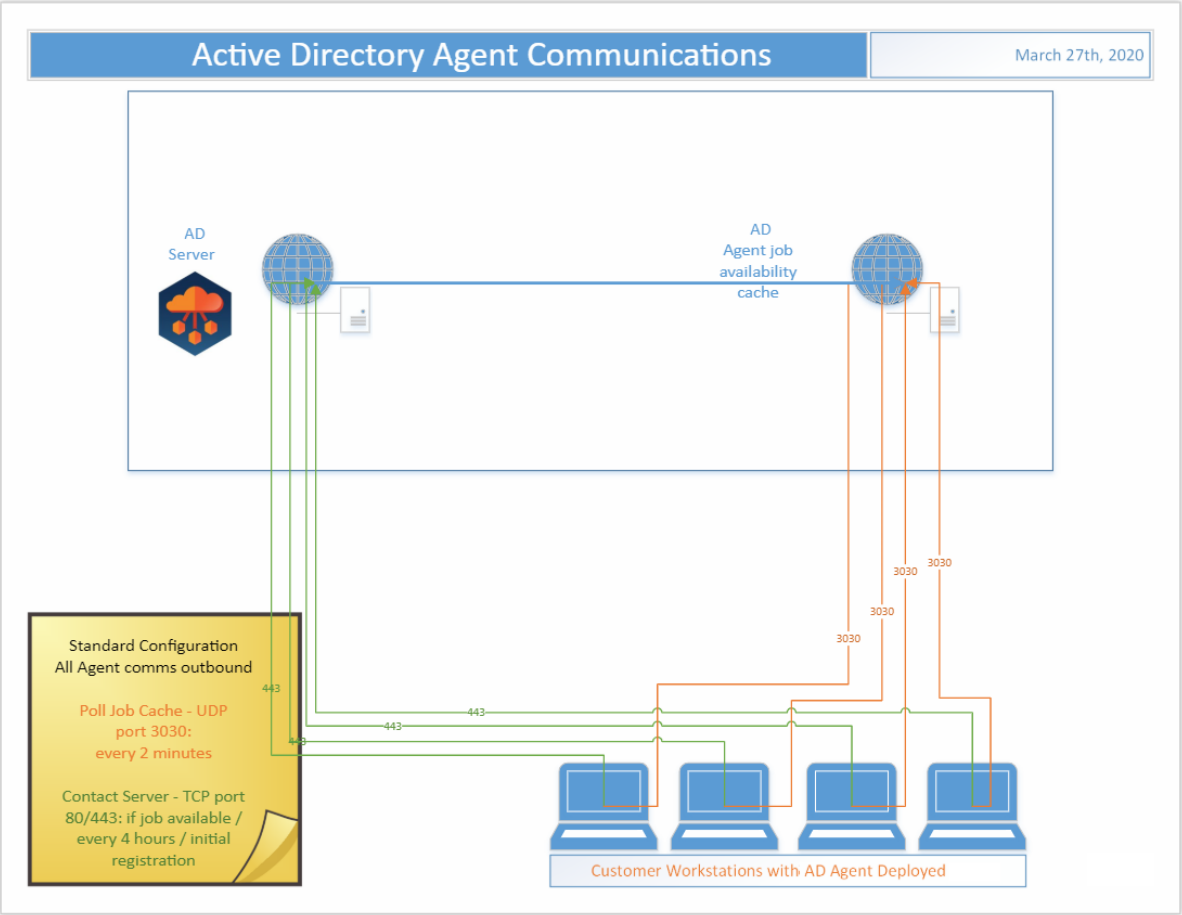
The directory synchronization engine, the web service, and the management interface will all access the same SQL database. In most scenarios, these components will be installed on the same system. In larger or more complex network environments, the components can be distributed across multiple systems.

User workstations, member servers and computers are collectively referred to as Devices in Entra ID for Devices. Computers communicate with the Entra ID for Devices web service using the Active Directory Agent. The Active Directory Agent is a lightweight application that installs as a service on Windows computers.

To ensure that no firewall exceptions are required, the web service does not “call” the Devices to be migrated. Instead, the Active Directory Agents contact the web service at defined polling intervals, using standard HTTPS or HTTP requests to collect jobs. Jobs include key tasks such as system discovery, updating the operating system, file system, and user profile permissions, and migrating the computer to the new domain.

Standard Configuration

In the Standard Configuration for Entra ID for Devices, the Agent is deployed to each Device to be migrated. Those Agents communicate outbound to the Active Directory webserver in Azure over ports 80/443 every 4 hours, when a job is available, or when initially registering. They also communicate outbound to the Active Directory Agent job availability cache in Azure over UDP on port 3030 every 2 minutes.



Web Proxy Configuration

In the Web Proxy Configuration for Entra ID for Devices, the Agent is deployed to each Device to be migrated and use of a web proxy is enabled. Those Agents communicate outbound through the defined proxy port to the Active Directory webserver in Azure over port 443 every 4 hours, when a job is available, or when initially registering. They also communicate outbound through the defined proxy port to the Active Directory Agent job availability cache in Azure on port 80 every 2 minutes.

Troubleshooting

A workstation that has been successfully cutover no longer responds to any additional jobs, such as Cleanup.

If a workstation that has been successfully cutover now fails to respond to any additional jobs, such as Cleanup, check the Application event log. If you see a "The remote name could not be resolved" error, this most likely means that the SRV record for the Entra ID for Devices Server can no longer be resolved due to a DNS lookup failure.

If you cannot "Ping" the Entra ID for Devices server from any other machines in the target domain, then you will need to remedy this on a more global scale, such as creating a conditional forwarder on the target machines' current DNS server pointing to the appropriate location.

If you are able to "Ping" the Entra ID for Devices server, then check the Network Profile that was used during the Cutover to verify that the DNS settings were correct in that profile.

Cutover Job Result Codes

Result Code	Error	Rollback Possible
1	Unidentified Error - PowerShell Command Error	No
2	Source Domain could not be contacted	No
4	Bad Source Credentials	No
8	Target Domain could not be contacted	No
16	Bad Target Credentials	No
32	Target DNS Server could not be contacted or could not resolve the target DNS domain	No
64	Change Obtain DNS by DHCP	
128	Set DNS Server IPs	
256	Set WINS Servers	
512	Register NIC with DNS	
1024	Clear DNS Suffix Search List / Set to use NIC	
2048	Set Alternate DNS Suffix List	
4096	Enable Dynamic DNS Registration	
8192	Set NIC Specific DNS Suffix	
16384	Domain Disjoin Failed	
32768	Domain Join Failed	
65536	Source domain name does not match the system's domain	No
131072	Computer Reboot failed	
262144	Target Domain Name could not be resolved via existing DNS, and new DNS Servers were not provided	No

NOTE: An odd numbered result code represents an error running the Cutover PowerShell script. The most common cause of an odd numbered result code during Cutover is that the computer either has no network card with a default gateway or more than one network card with a default gateway.

Result codes are additive. There are likely multiple errors if the result code is not represented in the table.

Upload Logs Result Codes

This table includes result codes for BT-UploadLogs PowerShell jobs.

Result Code	Error	Rollback Possible
32	(zip folder) could not be created.	No
64	Failed to Zip log files on computer.	No
128	Upload failed to contact the server. Please verify the URL (url) is correct and BITS is enabled.	No

CSV Import File Script

Import File Script Guide

```
### A.) AD Express mapper helper script

- script contains functions which helps user to convert the AD object search results
into .csv mapping files
- documentation will explain only the function that is correlated to AD Express mapping
for EntraId for Device

1. ADConvertToCsv function
- helps organize the search result into .csv list files or .csv mapping files
- called by adding to AD object search pipeline:

<AD object search command> | ADConvertToCsv
(AD object search commands are explained in section C.)

- will create .csv file to the current script directory from the search result
- if "-Path" is specified, will create file in the path (... | ADConvertToCsv -Path
"C:\Users\userx\objects.csv")
- if "-Append" switch is specified, will append existing file (must be called with -
Path)
- if "-EntraId" switch is specified, will create mapping file required by ADExpress
EntraId for Device
- supports powershell AD(on-prem) and microsoft graph (cloud) search command results
- does not support powershell AzureAD (cloud) search command results, because of
deprecated state and insufficient functionality

### B.) INSTRUCTIONS

1. prepare powershell window
-for getting AD on-prem objects (section C.1.)
    - available by default
    - any version (https://learn.microsoft.com/en-
us/powershell/module/activedirectory/?view=windowsserver2022-ps&source=recommendations)

-for getting AD cloud objects via microsoft graph (section C.2.)
    - https://learn.microsoft.com/en-
us/powershell/microsoftgraph/installation?view=graph-powershell-1.0
    - powershell version 5.1 and higher (powershell 7 recommended)
    - powershell 7 installation guide https://learn.microsoft.com/en-
us/powershell/scripting/install/installing-powershell-on-windows?view=powershell-7.4
```

```
2. Import script to the current powershell session
- run powershell
- move to directory where helper script is located (find directory/folder in UI ->
shift-rightmouseclick -> "copy as path"
"cd <directory_path>"
- load script
". .\BT-ADObjectMapper.ps1"

3. Create mapping file with source (on-prem) objects filled

-this will create .csv mapping file with source objects filled (on-prem) and empty
target objects
-this requires rights to search through on-prem AD
-function ADConvertToCsv will require switch "-Source" and "-EntraId" to be used
    - "-Source" specifies that we want to create mapping file and fill source objects
    - "-EntraId" specifies that we want to create mapping file for "ADExpress for
EntraId"
-use section C.1. to learn about AD on-prem object search commands
-when creating mapping file for DEVICEs, do not continue with step 4. and 5. The
mapping file for DEVICEs is final with step 3.

-structure:
<AD object search command> |  ADConvertToCsv -Source -EntraId

-example:
Get-ADUser -Filter * -SearchBase
"OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM"   |   ADConvertToCsv -Source -EntraId


4. Create object file with target (cloud) objects

-this step is required for USERs and GROUPs, not to be used for DEVICEs
-this will create standalone file with target objects that will need to be matched with
source objects in next step
-requires rights to search through cloud AD
-function ADConvertToCsv does not require any switch for this step
-use section C.2. to learn about AD cloud object search commands

-structure:
<AD cloud object search command> | ADConvertToCsv

-example:
Get-MgUser -All | ADConvertToCsv

5. Map source objects to target objects

-this step is required for USERs and GROUPs, not to be used for DEVICEs
-match objects in the mapping file the way you want
-need to manually map target objects from object file created in step 4 to the source
objects in the mapping file created in step 3
-copy respective attributes of object in the mapping file to columns prefixed with
"Target"
```

Import File Script Guide

```
### C.) AD Object search commands

#C.1. AD (on-premises)

C.1.0 Filtering Local objects
-Filter <query>
specifies query string that retrieves AD objects
-SearchBase <path>
specifies Active Directory path to search under

#examples:
Get-ADUser -Filter * -SearchBase "OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM"
-gets all users (specified by -Filter *) in the container
"OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM"

Get-ADUser -Filter 'SamAccountName -like "*test"' -SearchBase
"OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM"
-gets all users, which have SamAccountName ending with "test" in the container
"OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM"

*more in doc links

C.1.1 USER
function:
Get-AdUser (https://learn.microsoft.com/en-us/powershell/module/activedirectory/get-
aduser?view=windowsserver2022-ps)
example:
Get-ADUser -Filter * -SearchBase "OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM"|
ADConvertToCsv

C.1.2 GROUP
function:
Get-ADGroup (https://learn.microsoft.com/en-us/powershell/module/activedirectory/get-
adgroup?view=windowsserver2022-ps)
example:
Get-ADGroup -Filter * -SearchBase "OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM" |
ADConvertToCsv

C.1.3 DEVICE
function:
Get-ADComputer (https://learn.microsoft.com/en-
us/powershell/module/activedirectory/get-adcomputer?view=windowsserver2022-ps)
example:
Get-ADComputer -Filter * -SearchBase "OU=Finance,OU=UserAccounts,DC=FABRIKAM,DC=COM" |
ADConvertToCsv


#C.2 AD (cloud) using Microsoft Graph

C.2.0 Filtering Cloud objects using Microsoft Graph
before calling any function we need to call Connect-MgGraph to connect to the azure
```

Import File Script Guide

```
entra id (https://learn.microsoft.com/en-
us/powershell/module/microsoft.graph.authentication/connect-mggraph?view=graph-
powershell-1.0)

-All
returns all users

-Filter <filter>
allows for OData v3.0 filtering (https://www.odata.org/documentation/odata-version-3-
0/odata-version-3-0-core-protocol/#queryingcollections)

-Search <phrase>
search items by search phrases

-examples:
Get-MgUser -Filter "startsWith(DisplayName, 'a')"
-search users which displayname starts with "a" (case-insensitive)
Get-MgUser -Search '"Mail:Peter"'
-search users which email adress contains "Peter"

*more in doc links

C.2.1 USER
function:
Get-MgUser (https://learn.microsoft.com/en-
us/powershell/module/microsoft.graph.users/get-mguser?view=graph-powershell-
1.0&preserve-view=true)
example:
Connect-MgGraph -Scopes 'User.Read.All' (to get permissions and choose entra id-
interactive login)
Get-MgUser -All | ADConvertToCsv

C.2.2 GROUP
function:
Get-MgGroup (https://learn.microsoft.com/en-
us/powershell/module/microsoft.graph.groups/get-mggroup?view=graph-powershell-1.0)
example:
Connect-MgGraph -Scopes 'Group.Read.All' (to get permissions and choose entra id-
interactive login)
Get-MgGroup -All | ADConvertToCsv

C.2.3 DEVICE
function:
Get-MgDevice (https://learn.microsoft.com/en-
us/powershell/module/microsoft.graph.identity.directorymanagement/get-
mgdevice?view=graph-powershell-1.0)
example:
Connect-MgGraph -Scopes 'Device.Read.All' (to get permissions and choose entra id-
interactive login)
Get-MgDevice -All | ADConvertToCsv

*to switch entra id
-disconnect
```

Import File Script Guide

```
Disconnect-MgGraph
-connect to a different one
Connect-MgGraph -Scopes <required_scopes>
```

Import File Script

```
function ADConvertToCsv() #Used to convert AD(onprem/cloud) objects
(users,groups,devices/computers) to a .csv list for AD Express mapping file.
{
    Param(
        [Parameter(
            Mandatory=$true,
            ValueFromPipeline=$true)]
        $InputObject, #Can take input from pipeline

        [Parameter(Mandatory=$false)]
        [string]$Path, #Path to output .csv file, if not provided, output will be
created in current ps directory

        [Parameter(Mandatory=$false)]
        [switch]$Append, #Appends existing .csv file (-Path must be specified)

        [Parameter(Mandatory=$false)]
        [switch]$Source, #To create template mapping file with input as source objects

        [Parameter(Mandatory=$false)]
        [switch]$EntraId #To create mapping file for 'EntraID for Device' (can only be
used in conjunction with -Source switch)
    )
    Begin
    {
        #append or rewrite
        if ($Append)
        {
            if (!$Path)
            {
                Throw "[ERROR]: Path must be specified (-Path) in order to append
existing file."
            }
            elseif (!(Test-Path $Path))
            {
                Throw "[ERROR]: File does not exist. It needs to exist in order to be
appended."
            }

            $saveOperationType = @{ Append = $true }
        }
        else {
            $saveOperationType = @{ Force = $true }
        }
```

```
        #specify output path
        $outputPath = ""
        if ($Path)
        {
            $outputPath = $Path
        }
        else
        {
            if ($Source)
            {
                $outputPath = "$PSScriptRoot\object_mapping_file_" +
[DateTime]::Now.ToString('yyyy_MM_dd_HH_mm_ss') + ".csv"
            }
            else
            {
                $outputPath = "$PSScriptRoot\object_list_" + [DateTime]::Now.ToString
('yyyy_MM_dd_HH_mm_ss') + ".csv"
            }
        }

        if ($EntraId -and (-not $Source))
        {
            Throw "[ERROR]: Switch -EntraId can only be used in conjunction with -
Source switch."
        }

        $inputList = @()
        $inputType = ""
        $uniqueDomainList = @()

    }
    Process
    {
        if ($InputObject)
        {
            #make sure the type is allowed and all objects are of same type
            if ($inputType -eq "")
            {
                $inputType = $InputObject.GetType().FullName

                if ($EntraId) #switch only for on-prem objects for 'EntraID for Device'
                {
                    if ($inputType -notin @
("Microsoft.ActiveDirectory.Management.ADUser",
                                        "Microsoft.ActiveDirectory.Management.ADGroup",
                                        "Microsoft.ActiveDirectory.Management.ADComput
er"))
                    {
                        Throw "[ERROR]: Type $inputType is not one of the accepted
types."
                    }
                }
```

```
                else
                {
                    if ($inputType -notin @
("Microsoft.ActiveDirectory.Management.ADUser",
                                            "Microsoft.ActiveDirectory.Management.ADGr
oup",
                                            "Microsoft.ActiveDirectory.Management.ADCo
mputer",
                                            "Microsoft.Graph.PowerShell.Models.Microso
ftGraphUser",
                                            "Microsoft.Graph.PowerShell.Models.Microso
ftGraphGroup",
                                            "Microsoft.Graph.PowerShell.Models.Microso
ftGraphDevice",
                                            "Microsoft.Open.AzureAD.Model.User",
                                            "Microsoft.Open.AzureAD.Model.Group",
                                            "Microsoft.Open.AzureAD.Model.Device"))
                    {
                        Throw "[ERROR]: Type $inputType is not one of the accepted
types."
                    }
                }
            }
            elseif ($InputObject.GetType().FullName -ne $inputType)
            {
                Throw "[ERROR]: Type $($InputObject.GetType().FullName) is not the same
type as previous $inputType."
            }

            if ($EntraId) #get DC from distinguished name and get domain info
            {
                $dnList = $InputObject.DistinguishedName -split '(?<!\\),' | Where-
Object { $_ -like "DC*" }
                $dnDC = $dnList -join ","

                if ($uniqueDomainList.DNPath -notcontains $dnDC)
                {
                    $result = Get-ADDomain -Identity $dnDC
                    if (-not $result)
                    {
                        Throw "[ERROR]: Domain not found for object " +
$InputObject.Name
                    }

                    $domainObject = [PSCustomObject]@{
                        DNPath = $dnDC
                        NetBIOSName = $result.NetBIOSName
                        DomainName = $result.Name
                    }

                    $uniqueDomainList += $domainObject
                }
                else
                {
```

```
                    $domainObject = $uniqueDomainList | Where-Object {$_.DNPath -eq
$dnDC}
                }

                $InputObject | Add-Member -NotePropertyName NetBIOSName -
NotePropertyValue $domainObject.NetBIOSName -Force
                $InputObject | Add-Member -NotePropertyName DomainName -
NotePropertyValue $domainObject.DomainName -Force
            }

            $inputList += $InputObject
        }
    }
    End
    {

        if ($inputList.Count -eq 0)
        {
            Write-Host "[INFO]: There are no records. Csv file will not be created."
            return
        }

        #transform input into .csv file
        switch ($inputType)
        {
            "Microsoft.ActiveDirectory.Management.ADUser"
            {
                if ($Source)
                {
                    if ($EntraId)
                    {
                        $inputList |
                            Select-Object -Property @{Name="SourceObjectId";
Expression={$_.ObjectGUID}}, `
                                @{Name="SourceUserPrincipalName"; Expression=
{$_.UserPrincipalName}}, `
                                @{Name="SourceSamAccountName"; Expression=
{$_.SamAccountName}}, `
                                @{Name="SourceNetBIOSName"; Expression=
{$_.NetBIOSName}}, `
                                @{Name="SourceDomainName"; Expression={$_.DomainName}},
`
                                @{Name="SourceObjectSID"; Expression={$_.SID}}, `
                                @{Name="TargetObjectId"; Expression={""}}, `
                                @{Name="TargetUserPrincipalName"; Expression={""}}, `
                                @{Name="TargetSamAccountName"; Expression={""}}, `
                                @{Name="Comments (Optional)"; Expression={""}} |
                                    Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                    }
                    else
                    {
                        $inputList |
                            Select-Object -Property @{Name="SourceObjectId";
```

```
Expression={$_.ObjectGUID}}, `
                                @{Name="SourceUserPrincipalName (Optional)";
Expression={$_.UserPrincipalName}}, `
                                @{Name="SourceSamAccountName (Optional)"; Expression=
{$_.SamAccountName}}, `
                                @{Name="TargetObjectId"; Expression={""}}, `
                                @{Name="TargetUserPrincipalName (Optional)";
Expression={""}}, `
                                @{Name="TargetSamAccountName (Optional)"; Expression=
{""}}, `
                                @{Name="Comments (Optional)"; Expression={""}} |
                                    Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                    }
                }
                else
                {
                    $inputList |
                        Select-Object -Property @{Name="ObjectId"; Expression=
{$_.ObjectGUID}}, `
                            @{Name="UserPrincipalName"; Expression=
{$_.UserPrincipalName}}, `
                            @{Name="SamAccountName"; Expression={$_.SamAccountName}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
            }

            "Microsoft.ActiveDirectory.Management.ADGroup"
            {
                if ($Source)
                {
                    if ($EntraId)
                    {
                        $inputList |
                            Select-Object -Property @{Name="SourceObjectId";
Expression={$_.ObjectGUID}}, `
                                @{Name="SourceSamAccountName"; Expression=
{$_.SamAccountName}}, `
                                @{Name="SourceNetBIOSName"; Expression=
{$_.NetBIOSName}}, `
                                @{Name="SourceDomainName"; Expression={$_.DomainName}},
`
                                @{Name="SourceObjectSID"; Expression={$_.SID}}, `
                                @{Name="TargetObjectId"; Expression={""}}, `
                                @{Name="TargetSamAccountName"; Expression={""}}, `
                                @{Name="Comments (Optional)"; Expression={""}} |
                                    Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                    }
                    else
                    {
                        $inputList |
                            Select-Object -Property @{Name="SourceObjectId";
```

```
Expression={$_.ObjectGUID}}, `
                                @{Name="SourceSamAccountName (Optional)"; Expression=
{$_.SamAccountName}}, `
                                @{Name="TargetObjectId"; Expression={""}}, `
                                @{Name="TargetSamAccountName (Optional)"; Expression=
{""}}, `
                                @{Name="Comments (Optional)"; Expression={""}} |
                                    Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                    }
                }
                else
                {
                    $inputList |
                        Select-Object -Property @{Name="ObjectId"; Expression=
{$_.ObjectGUID}}, `
                            @{Name="SamAccountName"; Expression={$_.SamAccountName}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
            }
            "Microsoft.ActiveDirectory.Management.ADComputer" {
                if ($Source)
                {
                    if ($EntraId)
                    {
                        $inputList |
                            Select-Object -Property @{Name="SourceObjectId";
Expression={$_.ObjectGUID}}, `
                                @{Name="SourceSamAccountName"; Expression=
{$_.SamAccountName}}, `
                                @{Name="SourceDomainName"; Expression={$_.DomainName}},
`
                                @{Name="Comments (Optional)"; Expression={""}} |
                                    Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                    }
                    else
                    {
                        $inputList |
                            Select-Object -Property @{Name="SourceObjectId";
Expression={$_.ObjectGUID}}, `
                                @{Name="SourceSamAccountName (Optional)"; Expression=
{$_.SamAccountName}}, `
                                @{Name="Comments (Optional)"; Expression={""}} |
                                    Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                    }
                }
                else
                {
                    $inputList |
                        Select-Object -Property @{Name="ObjectId"; Expression=
{$_.ObjectGUID}}, `
```

```
                            @{Name="SamAccountName"; Expression={$_.SamAccountName}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
            }
            "Microsoft.Graph.PowerShell.Models.MicrosoftGraphUser"{
                if ($Source)
                {
                    $inputList |
                        Select-Object -Property @{Name="SourceObjectId"; Expression=
{$_.Id}}, `
                            @{Name="SourceUserPrincipalName (Optional)"; Expression=
{$_.UserPrincipalName}}, `
                            @{Name="SourceSamAccountName (Optional)"; Expression=
{$_.OnPremisesSamAccountName}}, `
                            @{Name="TargetObjectId"; Expression={""}}, `
                            @{Name="TargetUserPrincipalName (Optional)"; Expression=
{""}}, `
                            @{Name="TargetSamAccountName (Optional)"; Expression={""}},
`
                            @{Name="Comments (Optional)"; Expression={""}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
                else
                {
                    $inputList |
                        Select-Object -Property @{Name="ObjectId"; Expression={$_.Id}},
`
                            @{Name="UserPrincipalName"; Expression=
{$_.UserPrincipalName}}, `
                            @{Name="SamAccountName"; Expression=
{$_.OnPremisesSamAccountName}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
            }
            "Microsoft.Graph.PowerShell.Models.MicrosoftGraphGroup"{
                if ($Source)
                {
                    $inputList |
                        Select-Object -Property @{Name="SourceObjectId"; Expression=
{$_.Id}}, `
                            @{Name="SourceSamAccountName (Optional)"; Expression=
{$_.OnPremisesSamAccountName}}, `
                            @{Name="TargetObjectId"; Expression={""}}, `
                            @{Name="TargetSamAccountName (Optional)"; Expression={""}},
`
                            @{Name="Comments (Optional)"; Expression={""}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
                else
                {
                    $inputList |
```

```
                        Select-Object -Property @{Name="ObjectId"; Expression={$_.Id}},
`
                            @{Name="SamAccountName"; Expression=
{$_.OnPremisesSamAccountName}}, `
                            @{Name="DisplayName"; Expression={$_.DisplayName}}, `
                            @{Name="EmailAddress"; Expression={$_.Mail}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
            }
            "Microsoft.Graph.PowerShell.Models.MicrosoftGraphDevice"{
                if ($Source)
                {
                    $inputList |
                        Select-Object -Property @{Name="SourceObjectId"; Expression=
{$_.Id}}, `
                            @{Name="SourceSamAccountName (Optional)"; Expression=
{$_.OnPremisesSamAccountName}}, `
                            @{Name="Comments (Optional)"; Expression={""}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
                else
                {
                    $inputList |
                        Select-Object -Property @{Name="ObjectId"; Expression={$_.Id}},
`
                            @{Name="SamAccountName"; Expression=
{$_.OnPremisesSamAccountName}}, `
                            @{Name="DisplayName"; Expression={$_.DisplayName}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
            }
            "Microsoft.Open.AzureAD.Model.User" {
                if ($Source)
                {
                    $inputList |
                        Select-Object -Property @{Name="SourceObjectId"; Expression=
{$_.ObjectId}}, `
                            @{Name="SourceUserPrincipalName (Optional)"; Expression=
{$_.UserPrincipalName}}, `
                            @{Name="SourceSamAccountName (Optional)"; Expression={""}},
`
                            @{Name="TargetObjectId"; Expression={""}}, `
                            @{Name="TargetUserPrincipalName (Optional)"; Expression=
{""}}, `
                            @{Name="TargetSamAccountName (Optional)"; Expression={""}},
`
                            @{Name="Comments (Optional)"; Expression={""}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
                else
                {
```

```
                    $inputList |
                        Select-Object -Property @{Name="ObjectId"; Expression=
{$_.ObjectId}}, `
                            @{Name="UserPrincipalName"; Expression=
{$_.UserPrincipalName}}, `
                            @{Name="SamAccountName"; Expression={""}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
            }
            "Microsoft.Open.AzureAD.Model.Group" {
                if ($Source)
                {
                    $inputList |
                        Select-Object -Property @{Name="SourceObjectId"; Expression=
{$_.ObjectId}}, `
                            @{Name="SourceSamAccountName (Optional)"; Expression={""}},
`
                            @{Name="TargetObjectId"; Expression={""}}, `
                            @{Name="TargetSamAccountName (Optional)"; Expression={""}},
`
                            @{Name="Comments (Optional)"; Expression={""}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
                else
                {
                    $inputList |
                        Select-Object -Property @{Name="ObjectId"; Expression=
{$_.ObjectId}}, `
                            @{Name="SamAccountName"; Expression={""}}, `
                            @{Name="DisplayName"; Expression={$_.DisplayName}}, `
                            @{Name="EmailAddress"; Expression={$_.Mail}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
            }
            "Microsoft.Open.AzureAD.Model.Device" {
                if ($Source)
                {
                    $inputList |
                        Select-Object -Property @{Name="SourceObjectId"; Expression=
{$_.ObjectId}}, `
                            @{Name="SourceSamAccountName (Optional)"; Expression={""}},
`
                            @{Name="Comments (Optional)"; Expression={""}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
                else
                {
                    $inputList |
                        Select-Object -Property @{Name="ObjectId"; Expression=
{$_.ObjectId}}, `
```

```
                            @{Name="SamAccountName"; Expression={""}}, `
                            @{Name="DisplayName"; Expression={$_.DisplayName}} |
                                Export-Csv -Path $outputPath -Delimiter "," -
NoTypeInformation @saveOperationType
                }
            }
            default {
                Throw "[ERROR]: Type $inputType is not one of the accepted types."
            }
        }

        if ($Append)
        {
            Write-Host "[INFO]: File '$($outputPath)' appended"
        }
        else
        {
            Write-Host "[INFO]: File '$($outputPath)' created"
        }
    }
}


function ADMapLocalUsers($sourceUsersPath, $targetUsersPath) #Map two user files based
on SamAccountName
{
    $sourceUsers = Import-Csv -Path $sourceUsersPath -Delimiter ","
    $targetUsers = Import-Csv -Path $targetUsersPath -Delimiter ","
    $mappedUsers = @()
    $nonMappedUsers = @()

    #for each source user, try to find target user via SamAccountName
    $sourceUsers | % {
        $sourceObject = $_
        $targetMatch = $null
        if ($sourceObject.SamAccountName -ne "")
        {
            $targetMatch = $targetUsers | Where-Object {$_.SamAccountName -eq
$sourceObject.SamAccountName}
        }

        if ($targetMatch) #if found, create match record
        {
            if ($targetMatch.Count -gt 1)
            {
                throw "[ERROR]: Target User SamAccountName is not unique
'$($sourceObject.SamAccountName)'"
            }

            $mappedUsers += [PSCustomObject]@{
                SourceObjectId = $sourceObject.ObjectId
                SourceUserPrincipalName = $sourceObject.UserPrincipalName
                SourceSamAccountName = $sourceObject.SamAccountName
```

```
                TargetObjectId = $targetMatch.ObjectId
                TargetUserPrincipalName = $targetMatch.UserPrincipalName
                TargetSamAccountName = $targetMatch.SamAccountName
                Comments = ""
            }
        }
        else #if havent found, create record without
        {
            $nonMappedUsers += [PSCustomObject]@{
                SourceObjectId = $sourceObject.ObjectId
                SourceUserPrincipalName = $sourceObject.UserPrincipalName
                SourceSamAccountName = $sourceObject.SamAccountName
                TargetObjectId = ""
                TargetUserPrincipalName = ""
                TargetSamAccountName = ""
                Comments = ""
            }
        }
    }

    #save target users which dont have source user match
    $sourceUsersWithSam = $sourceUsers | Where-Object {$_.SamAccountName -ne ""}
    $targetUsersWithoutSourceSam = $targetUsers | Where-Object { $_.SamAccountName -
notin $sourceUsersWithSam.SamAccountName}
    $targetUsersWithoutSourceSam | % {
        $nonMappedUsers += [PSCustomObject]@{
            SourceObjectId = ""
            SourceUserPrincipalName = ""
            SourceSamAccountName = ""
            TargetObjectId = $_.ObjectId
            TargetUserPrincipalName = $_.UserPrincipalName
            TargetSamAccountName = $_.SamAccountName
            Comments = ""
        }
    }

    #save
    $mappedUsers += $nonMappedUsers
    $outputPath = "$PSScriptRoot\user_map_" + [DateTime]::UtcNow.ToString('yyyy_MM_dd_
HH_mm_ss') + ".csv"
    $mappedUsers | Export-Csv -Path $outputPath -Delimiter "," -NoTypeInformation
    Write-Host "[INFO]: File '$($outputPath)' created"
}
function ADMapLocalGroups($sourceGroupsPath, $targetGroupsPath) #Map two group files
based on SamAccountName
{
    $sourceGroups = Import-Csv -Path $sourceGroupsPath -Delimiter ","
    $targetGroups = Import-Csv -Path $targetGroupsPath -Delimiter ","
    $mappedGroups = @()
    $nonMappedGroups = @()

    #for each source group, try to find target group via SamAccountName
    $sourceGroups | % {
```

```
        $sourceObject = $_
        $targetMatch = $null
        if ($sourceObject.SamAccountName -ne "")
        {
            $targetMatch = $targetGroups | Where-Object {$_.SamAccountName -eq
$sourceObject.SamAccountName}
        }

        if ($targetMatch) #if found, create match record
        {
            if ($targetMatch.Count -gt 1)
            {
                throw "Target Group SamAccountName is not unique
'$($sourceObject.SamAccountName)'"
            }

            $mappedGroups += [PSCustomObject]@{
                SourceObjectId = $sourceObject.ObjectId
                SourceSamAccountName = $sourceObject.SamAccountName
                TargetObjectId = $targetMatch.ObjectId
                TargetSamAccountName = $targetMatch.SamAccountName
                Comments = ""
            }
        }
        else #if havent found, create record without
        {
            $nonMappedGroups += [PSCustomObject]@{
                SourceObjectId = $sourceObject.ObjectId
                SourceSamAccountName = $sourceObject.SamAccountName
                TargetObjectId = ""
                TargetSamAccountName = ""
                Comments = ""
            }
        }
    }

    #save target groups which dont have source group match
    $sourceGroupsWithSam = $sourceGroups | Where-Object {$_.SamAccountName -ne ""}
    $targetGroupsWithoutSourceSam = $targetGroups | Where-Object { $_.SamAccountName -
notin $sourceGroupsWithSam.SamAccountName}
    $targetGroupsWithoutSourceSam | % {
        $nonMappedGroups += [PSCustomObject]@{
            SourceObjectId = ""
            SourceSamAccountName = ""
            TargetObjectId = $_.ObjectId
            TargetSamAccountName = $_.SamAccountName
            Comments = ""
        }
    }

    #save
    $mappedGroups += $nonMappedGroups
    $outputPath = "$PSScriptRoot\group_map_" + [DateTime]::UtcNow.ToString('yyyy_MM_dd_
HH_mm_ss') + ".csv"
```

```
    $mappedGroups | Export-Csv -Path $outputPath -Delimiter "," -NoTypeInformation
    Write-Host "[INFO]: File '$($outputPath)' created"
}
```

About us

Quest Software creates technology and solutions that build the foundation for enterprise AI. Focused on data management and governance, cybersecurity and platform modernization, Quest helps organizations address their most pressing challenges and make the promise of AI a reality. Around the globe, more than 45,000 companies including over 90% of the Fortune 500 count on Quest Software. For more information, visit www.quest.com or follow Quest Software on X (formerly Twitter) and LinkedIn.

Technical support resources

Technical support is available to Quest customers with a valid maintenance contract and customers who have trial versions. You can access the Quest Support Portal at https://support.quest.com.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request
- View Knowledge Base articles
- Sign up for product notifications
- Download software and technical documentation
- View how-to-videos
- Engage in community discussions
- Chat with support engineers online
- View services to assist you with your product