



# BeyondTrust

## **Privileged Identity 7.3 Disconnected Account Management**

## Table of Contents

---

<b>Privileged Identity Disconnected Account Management</b> .....	<b>3</b>
<b>Set Up Disconnected Account Management</b> .....	<b>4</b>
Configure the Server for Disconnected Account Management .....	5
Create Disconnected Account Lists .....	7
Manage Password Policies for Disconnected Accounts .....	9
Delegate Permissions for Disconnected Account Access .....	14
Enroll Endpoints in a Disconnected Account List .....	16
Configure Endpoint Clients .....	19
<b>Access Disconnected Passwords and Elevate Disconnected Accounts</b> .....	<b>27</b>
<b>Review Logs for Disconnected Account Activity</b> .....	<b>31</b>
<b>Remove Disconnected Account Lists and Clients</b> .....	<b>32</b>
Remove Clients from Lists .....	32
Remove Lists .....	32
<b>Notes about Disconnected Account Management</b> .....	<b>34</b>

## Privileged Identity Disconnected Account Management

Privileged Identity helps you to establish a base of knowledge regarding the systems and devices in your network, to identify what accounts are on those systems and devices, and to enable ongoing password or SSH key rotation for those accounts. For proactive discovery and management of systems, the target systems need to be online and have network connectivity with Privileged Identity. However, many users work offline on a regular basis, making this proactive management difficult at best.

To solve this problem, we offer disconnected account management. Disconnected account management (DAM) allows you to continue password randomization on systems that do not regularly connect to the network. With the elevation feature, privileged users can have access to offline systems with the confidence that the admin account will have continued randomized passwords on a scheduled basis.

Disconnected account management generates cryptographically secure secret data on the server that hosts the web service. Machines managed by DAM never really need to connect to the network. They must, however, establish a connection to the web service on a scheduled basis.

Each client endpoint pulls a shared secret from the web service, as well as password policy settings. Both the clients and the server use the same one-way hashing algorithms and the same secret data to derive a series of passwords for local accounts. Because the endpoints and the server share the same settings, they can both at any time calculate the derived current password from the known secret, even if they are not connected.

DAM never stores passwords. Instead, it regularly derives new passwords from the cryptographically secure secret data. You can set the criteria for this data, determining the frequency of new passwords, custom characters, password length, and frequency of shared secret changes. When scheduled, the web service updates the shared secret on the DAM-managed system, and the cycle of deriving a new password repeats. This process helps mitigate the risk of pass-the-hash attacks, and it helps to avoid lateral escalation scenarios.



### **IMPORTANT!**

*Disconnected account management is a licensed feature of Privileged Identity. To purchase a DAM license, please contact your BeyondTrust Sales rep.*

## Set Up Disconnected Account Management

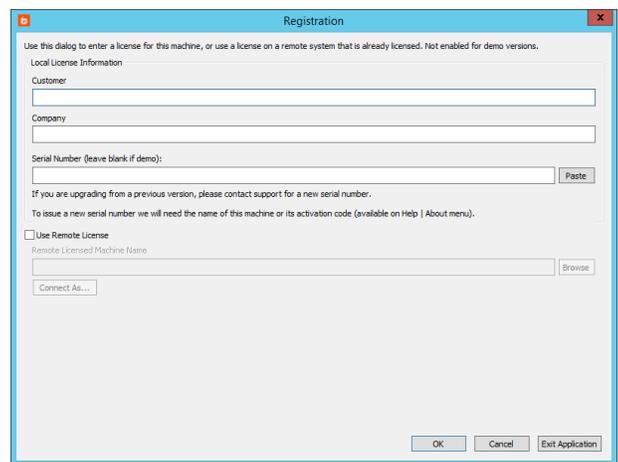
Before disconnected account management (DAM) can begin managing offline systems, you must enable it in the web application settings, either in the web app itself or in its properties in the management console. This setup defines the initial policy. You can define further policies for each disconnected systems list.

Give permission to users who are allowed to use DAM. Download and install the client software on each system you want to manage. As part of the initial setup, the machines under management must connect to and download policy information from the web service host.

### Register Disconnected Account Management

To register your DAM license, follow these steps:

1. Open the Privileged Identity management console.
2. Select **Help > Register**.
3. In **Serial Number**, enter the DAM license key that you received from BeyondTrust Support.
4. Click **OK**.



## Configure the Server for Disconnected Account Management

On the server where you'll set up disconnected account management (DAM), make sure you have the Privileged Identity management console and the web service installed. These components hold the protected secret data for endpoints and distribute the shared secrets. You can manage the disconnected password recovery portal through the web interface or web service.

**i** The DAM web service is installed as part of the web service installation. For more information, please see [Install the Web Service](https://www.beyondtrust.com/docs/privileged-identity/install/install-software/web-service.htm) at <https://www.beyondtrust.com/docs/privileged-identity/install/install-software/web-service.htm>.

After you've installed the DAM web service, you can access the DAM web service on its host at:

```
https://serverName/ERPWebService/OfflineUpdateWebService.svc
```

### **!** IMPORTANT!

You must enable the web service host for anonymous authentication. If the web service is configured for integrated authentication or certificate authentication, the DAM feature will not work. If needed, you can make a copy of the web service files in a new directory and publish it in IIS with anonymous authentication. Be sure to configure the correct web service **web.config** file for anonymous authentication.

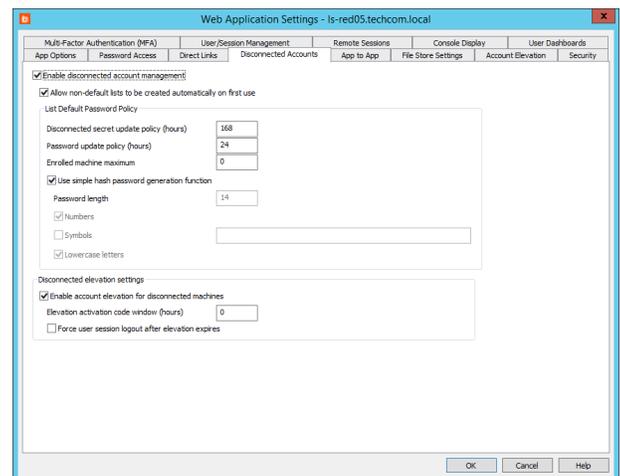
To enable disconnected account management, open the Privileged Identity management console and follow these steps:

1. Select **Manage Web App** from the left action pane.
2. Select the web application instance where you want to enable DAM, and click **Edit**.
3. Click **Yes** to overwrite the current settings.
4. Select the **Disconnected Accounts** tab.
5. Check **Enable disconnected account management**.
6. **Allow non-default lists to be created automatically on first use:** If you select this option, then if a new endpoint attempts to enroll with a list ID that does not yet exist, a new list is automatically created with the endpoint added as a member.

### List Default Password Policy

7. **Disconnected secret update policy:** Set how often a new secret should be generated on the server. The next time the managed system connects to the web service, if the secrets are mismatched, the managed system receives the latest secret.
8. **Password update policy:** Set how often a new derived password should be generated.
9. **Enrolled machine maximum:** Set the number of machines that can derive passwords from one secret. It is a best practice to divide systems into smaller management sets when dealing with large networks.
10. **Use simple hash password generation function:** If you select this option, then a password derived from the client secret will be a 14-character, random string of uppercase letters, lowercase letters, and numbers.

If you do not select this option, then set the following options for passwords derived from the client secret:



- **Password length:** Set how many characters to include in the password. The maximum is 127 characters.
- **Numbers:** Set if the password can contain numbers.
- **Symbols:** Set if the password can contain special characters. You can leave the text field blank to allow all possible symbols, or you can define an allowed list of symbols.



**Note:** To avoid causing code issues, you may not specify a slash (/), backslash (\), colon (:), semicolon (;), or quotation mark (").

Some databases accept only the special characters hash (#), underscore (\_), and dollar sign (\$)

- **Lowercase letters:** Set if the password can contain lowercase letters.

## Disconnected Elevation Settings

11. **Enable account elevation for disconnected machines:** If you select this option, you can elevate a user account to a predefined group on an offline, managed machine. The managed machine does not have to join the network to allow the user to be elevated.
12. **Elevation activation code window:** Set how long an elevation code should last.



**Note:** When you check out a code, the code remains valid for the length of time set here, plus the number of minutes remaining until the top of the hour. For example, if this is set to one hour and if you check out a code at 8:15, the code will remain valid until 10:00. You can reuse that same code any number of times within this time frame. If you need to elevate the account after this code has expired, you must go through the account elevation process again.

This code window is not related to how long an elevation lasts. The elevation window is set at the endpoint client. As long as the elevation begins while the code is valid, the elevation can continue past the code expiration.

13. **Force user session logout after elevation expires:** After an account has been de-elevated, you can force the end user to be logged off the managed system.
14. Once you've finished configuring these settings, click **OK**.

This configuration is the global default. To access the global policy settings in the web app, select **Settings > Site Settings**, and then scroll down to the **Disconnected Account Management** section.

In the web app, you can modify this configuration per list. Doing so overrides the defaults and applies the new settings to the specific list and all machines under that list. You may also modify the configuration per endpoint system.

All settings are stored on the server and updated to the managed node when the node connects to the web service.

## Create Disconnected Account Lists

After you've turned on disconnected account management (DAM), a user with the **All Access** permission can begin the process of managing offline Windows systems. This user can assign delegation to other users to help administer these machines.

Disconnected systems are broken into lists, each with a separate set of secrets and, if needed, a separate set of policies. You can create lists of disconnected systems directly through the web app.

 **Note:** If the setting **Allow non-default lists to be created automatically on first use** is enabled, then if a new endpoint attempts to enroll with a list ID that does not yet exist, a new list is automatically created with the endpoint added as a member.

When setting up a machine to manage with DAM, that system must be online and connected to the web service. After setup is complete and DAM is properly functioning, that system can disconnect until the next update cycle for the shared secret. This update cycle is set by either the global or list-specific policy.

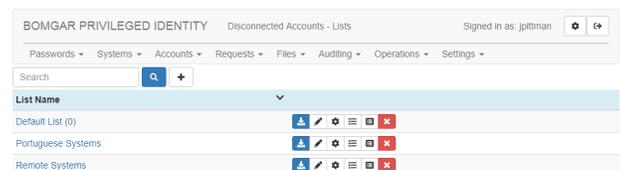
Each per-list policy is transferred with the DAM agent to target systems, and those systems become members of the list. You must manually push out any policy changes. When a secret updates, it is pushed to a managed endpoint the next time that endpoint connects to the web service.

 **Note:** To continue with DAM setup, you must have the web app installed and an all-access user delegated, and you must have a good understanding of management sets. A DAM list is very much like a management set and should be treated as such.

 For more information about management sets or user permissions, please see the [Privileged Identity Admin Guide](https://www.beyondtrust.com/docs/privileged-identity/documents/pi-admin.pdf) at [www.beyondtrust.com/docs/privileged-identity/documents/pi-admin.pdf](https://www.beyondtrust.com/docs/privileged-identity/documents/pi-admin.pdf).

### Create a List through the Web Application

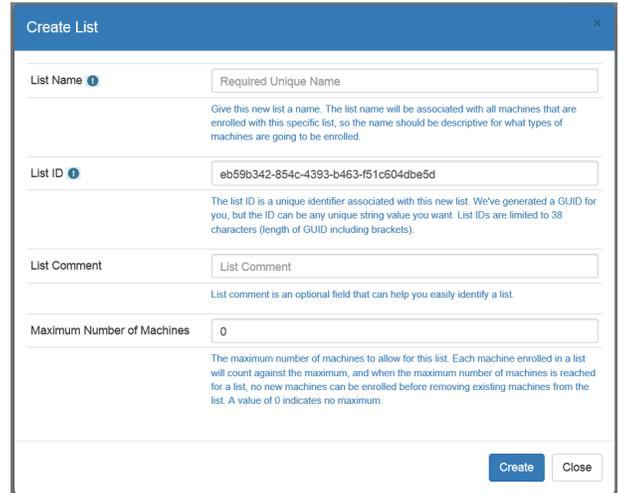
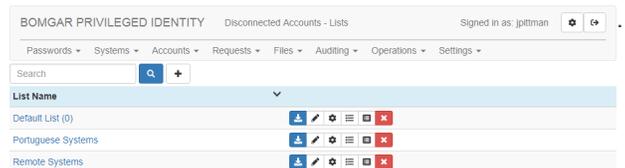
1. Log into the web application as a user with appropriate permissions.
2. Go to **Passwords > Disconnected Accounts**.
3. If this is your first list, a dialog automatically appears. Otherwise, click the **Create New List** button (+) near the top left.



4. In the **Create List** dialog:
  - a. **List Name:** Enter a descriptive name for the list.
  - b. **List ID:** While we highly recommend using the automatically generated ID, you may use any string up to 255 characters if you prefer more human-readable information.
  - c. **List Comment:** (*Optional*) Enter a comment or note for the list.
  - d. **Maximum Number of Machines:** Enter the maximum number of endpoints that can be added to this list. A value of **0** indicates an unlimited number of clients.

 **Note:** Once you've created a list, you cannot modify its ID.

5. Click **Create**.
6. You should now see the list you just created. You can modify its properties by clicking the **Edit List** button (pencil).

List Name	Actions
Default List (0)	[Edit] [Delete] [Refresh]
Portuguese Systems	[Edit] [Delete] [Refresh]
Remote Systems	[Edit] [Delete] [Refresh]

## Create a List Programmatically

Using the REST API, call `/REST/OfflineUpdate/Tenant` (POST).

## Manage Password Policies for Disconnected Accounts

When you enable disconnected account management (DAM), a default password policy is defined. You can also define password policies on a per-list basis. If no list-specific policy has been created for a list, clients of the list will use the default policy.

### Understand How Policies Affect Derived Password Generation

Once a secret has been generated and stored on the endpoint, it can begin deriving passwords. If a password set time has not been saved or if the password expiration time has elapsed, a new password will be derived from the stored secret and set for the local account. The password expiration time is calculated based on when the secret was obtained from the server and the password update frequency defined in the password policy.

Because the secret is generated on the server and saved on the client, the secret set time is always known by both systems. Therefore, the set times on the endpoint cannot drift over extended operation. Even if the update process cannot run for an extended length of time, such as when an endpoint is shut down, the password will be changed immediately when the update process is re-enabled, and the password will be re-synchronized with the value derived on the server. The endpoint also attempts to update its information on the server each time a new secret is obtained or when a new password is derived and set. However, since we expect the system to be disconnected, we also expect this update to be infrequent.

The algorithm used to derive the password from the stored secret depends on the password policy settings. Two types of password policies can be applied globally and/or per list:

- Simple hashing password generation
- Admin-defined password generation

Password policies let you configure the endpoint secret and password generation interval, set a wild card matching string to determine which endpoints are affected by each policy, and define the format for derived passwords. The password policy is copied to the endpoint as part of the endpoint enrollment process, and the policy is updated each time the endpoint obtains a new secret.

The default password policy is defined on the server as part of the web app settings. Unless otherwise specified, the default policy generates a new secret every 7 days and a new password every 24 hours, using the simple hashing algorithm.

Each endpoint will use exactly one policy. If an endpoint matches more than one policy, the first match is applied.

The endpoint determines when a new secret is required based on when the existing secret was generated and the current time on the endpoint. The endpoint determines when a new password is needed based on the last time the password was set and the current time. A new password is always set immediately when a new secret is generated and stored on an endpoint.

### Create a List Policy

You can create list policies either from the web application or programmatically.

#### Create a List Policy From the Web Application

1. Log into the web application as an **All Access** user.
2. Select **Passwords > Disconnected Accounts**.
3. Click the **List Policies** button (gear) in the list row.
4. Click the **Create New Policy** button (+) at the top of the page.

5. Define the following policy elements:

- **Machine Type Filter:** Enter a string to specify the types of systems to which this policy should apply. If you set it to an asterisk (\*), it will apply to all systems enrolled in the list.



For more information about filters, please see "Apply Policies Based on Machine Type" on page 10.

- **Secret Update Frequency:** Set how often a new secret should be generated on the server. The default is 168 hours (7 days). The next time the managed system connects to the web service, if the secrets are mismatched, the managed system receives the latest secret.
- **Password Update Frequency:** Set how often a new derived password should be generated. The default is 24 hours.
- **Simple Hash:** If you select this option, then a password derived from the client secret will be a 14-character, random string of uppercase letters, lowercase letters, and numbers.

If you do not select this option, then set the following options for passwords derived from the client secret:

- **Password length:** Set how many characters to include in the password. The maximum is 127 characters.
- **Numbers:** Set if the password can contain numbers.
- **Symbols:** Set if the password can contain special characters. You can leave the text field blank to allow all possible symbols, or you can define an allowed list of symbols.



**Note:** To avoid causing code issues, you may not specify a slash (/), backslash (\), colon (:), semicolon (;), or quotation mark (").

Some databases accept only the special characters hash (#), underscore (\_), and dollar sign (\$)

- **Lowercase letters:** Set if the password can contain lowercase letters.

6. Click **Create**.

## Apply Policies Based on Machine Type

The machine type filter is a wild card string that sets the policy to be used for any client systems with a matching machine type. The machine type identifies the endpoint's platform and system type. This string is generated by the endpoint itself when the client script runs. The Windows service and Python script generate machine type information in different formats:

OS Version	Windows Script Machine Type String	Python Script Machine Type String
Windows Vista	Microsoft Windows NT 6.0.6000	Vista-6.0.*
Windows Server 2008	Microsoft Windows NT 6.0.*	Vista-6.0.*
Windows Server 2008 R2	Microsoft Windows NT 6.1.*	Windows-7-6.1.*
Windows 7	Microsoft Windows NT 6.1.7600	Windows-7-6.1.*
Windows Server 2012	Microsoft Windows NT 6.2.*	Windows-post2008Server-6.2.*
Windows 8	Microsoft Windows NT 6.2.*	Windows-post2008Server-6.2.*
Windows Server 2012 R2	Microsoft Windows NT 6.3.*	Windows-post2008Server-6.2.*
Windows 8.1	Microsoft Windows NT 6.3.9600	Windows-post2008Server-6.2.*

OS Version	Windows Script Machine Type String	Python Script Machine Type String
Windows Server 2016	Microsoft Windows NT 10.0.*	Windows-post2008Server-6.2.*
Windows 10	Microsoft Windows NT 10.0.10240	Windows-post2008Server-6.2.*
SLES 12		Linux-4.4.21-69-default-x86_64-with-SuSE-12-x86_64
Raspberry Pi (Debian)		Linux-4.4.21-v7+-armv7l-with-debian-8.0
OSX (Sierra)		Darwin-13.4.0-x86_64-i386-64bit



**Note:** In the examples above, the specific patch level is replaced with an asterisk (\*), because the exact number varies.

While not all versions and distributions are shown, this table should provide a guide to relative formats. If you have a question about a specific format, you can run the endpoint client and observe what it reports. With the Python client, you can also start a Python session and run:

```
import platform
print platform.platform()
```

## Examine Password Generation Settings

### Simple Hashing Password Generation

The default derived password generation algorithm uses an MD5 hash combined with simple transforms to derive passwords with a simple, fixed format.

The first time a password is generated, the secret is hashed using MD5 to ensure good data distribution for the resulting hash. The resulting 64 bytes of binary data are interpreted as 32 hex-encoded characters. The first two hex digits are replaced with a question mark (?) to make sure the password contains a special character. The next fourteen characters are lowercase letters and numbers, and the remaining sixteen characters are uppercase letters and numbers.

For each following password generation, the previous derived password is used to generate a new MD5 hash (similar to how the secret is used the first time), and the same password derivation algorithm is applied to that hash.

Below is an example of a simple hashing password generation over two cycles:

Secret	pFKiZUQFIi3yJi4H1UvEb3gTRbADrV7E
MD5 Hash	
MD5 hash for first cycle	359df6d30033ed522c057ce5b5bdda96
Transform	
First password	?9df6d30033ed522C057CE5B5BDDA96
MD5 Hash	
MD5 hash for second cycle	964d3ffe07680f9b1696543c1dc634d8
Transform	
Second password	?4d3ffe07680f9b1696543C1DC634D8

## Defined Policy Password Generation

Derived password generation for a defined password policy is similar to the default method.

The first time a password is generated, the secret is hashed using MD5 to ensure good data distribution for the resulting hash. The resulting 64 bytes of binary data are interpreted as 32 hex-encoded characters. The process calculates the set of possible ASCII characters the resulting password can contain. In total, 95 characters are possible:

26 uppercase (always used)	ABCDEFGHIJKLMNOPQRSTUVWXYZ
26 lowercase	abcdefghijklmnopqrstuvwxyz
10 numbers	0123456789
33 symbols	[space] , . < > / ? ; : ' " [ ] { } \   ` ~ ! @ # \$ % ^ & * ( ) - _ = +

The set of allowed characters is constructed by adding all uppercase letters, all lowercase letters (if allowed), all numbers (if allowed), and all symbols (or the allowed symbols in the order they appear in the allowable symbols string). For each type of character that is not allowed, the size of the character set decreases. If a specified symbol set is used, the set size is decreased by the number of symbols that are not allowed.

Each two bytes of the binary data are interpreted as a decimal number between 1 and 255. That number is divided (mod) by the size of the allowable character set to get an index in the array of possible characters. One hash can generate a password up to sixteen characters. If the policy requires a longer password, another MD5 hash is generated from the first portion of the derived password, and that new hash is used to generate more characters, using the same method as before.

The remainder of this section looks at an example policy with a length of twenty characters, using uppercase and lowercase letters and numbers but no symbols:

Index Value	Character Result
0	A
1	B
...	...
26	a
27	b
...	...
52	0
53	1
...	...
60	8
61	9

Below is an example of conversion from a 2-byte hash value to a password character:

Machine secret	BQgApSMIqsHBD3xUIK1vLbtHb2uo6Gr3
MD5 Hash	
MD5 hash	a60b978fc6d0364de97aa218d0b15272
First 2 bytes	a6 (166 in decimal)
Character set size	95-33=62 (subtract symbols)

Character index mod	166 % 62 = 42
Character at index 42 in character set	q

Below is an example of a defined policy password generation over two cycles:

Secret	BQgApSMlqsHBD3xUIK1vLbtHb2uo6Gr3
MD5 Hash	
MD5 hash for first cycle	a60b978fc6d0364de97aa218d0b15272
Convert to password characters	
First 16 characters of first password	qLbTMW2Pv8mYW1U0
MD5 Hash first 16 characters	
MD5 hash from first 16 characters	70f18a8ff78f83ab64124276af3aad98
Convert to password characters	
Remaining 4 characters of first password	y3OT
Append results for password	
First password (16 + 4 characters)	qLbTMW2Pv8mYW1U0y3OT
MD5 Hash	
MD5 hash for second cycle	2977bfc0f1fb2fec8f8b7e501b97388d
Convert to password characters	
First 16 characters of second password	p5FG3DvyTPCSbb4R
MD5 Hash first 16 characters	
MD5 hash from first 16 characters	ae0b4ad18c6503fd7ed243cbc57c0566
Convert to password characters	
Remaining 4 characters of second password	yLMX
Append results for password	
Second password (16 + 4 characters)	p5FG3DvyTPCSbb4RyLMX

## Delegate Permissions for Disconnected Account Access

To retrieve passwords for or to elevate to disconnected systems, you must create lists of delegated users. You can delegate permissions through the web app or the management console. You can grant permissions either before or after you enroll disconnected systems.



**Note:** Permissions are granted on a per-list basis rather than a per-machine basis.



For more information on enrolling systems, please see ["Enroll Endpoints in a Disconnected Account List"](#) on page 16.

All-access users can automatically retrieve any password. Other users to require access to disconnected passwords must be granted permission. Delegated users can see only the lists to which they have been granted access. Lists must already exist before permissions may be granted.

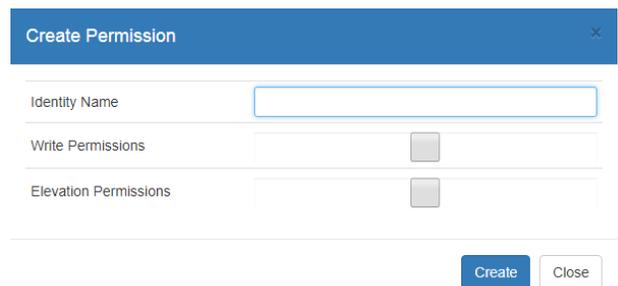
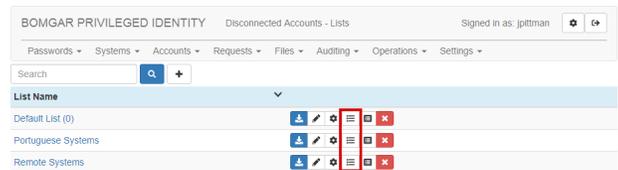
Delegating access to systems under disconnected account management (DAM) is separate from the normal delegation model. DAM delegation must be performed in the web app or management console by a user with the **All Access** permission.



**Note:** While DAM delegation does not fall under global delegation rules and permissions, a DAM-delegated identity still requires standard login permissions to the web app and must, therefore, be in the global delegation list.

### Grant Permissions through the Web App

1. Log into the web app with an all-access user account.
2. Go to **Passwords > Disconnected Accounts**.
3. Click the **List Permissions** button next to the list you want to manage.
4. If this is your first identity for this list, a dialog automatically appears. Otherwise, click the **Create New List** button (+) near the top left.
5. Enter an **Identity Name** to add to this list.



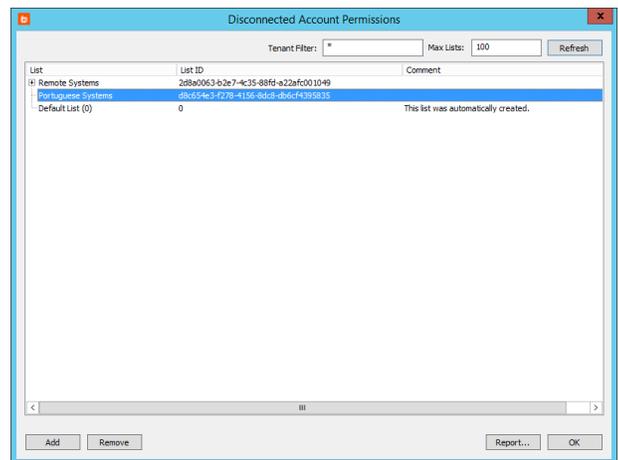
**Note:** To prevent slow web performance, you cannot look up names. Instead, you must manually enter the identity exactly as it appears in the web application **Settings > Delegation** list. For example, **domain/admin**, **jsmith**, or **Administrator User**.

6. Check **Write Permissions** if this identity should be allowed to modify settings associated with this list. If you leave this unchecked, this identity can only see this list's enrolled systems and their passwords. Write permissions include:
  - Creating and changing password policies for the list
  - Viewing endpoint machine and password information for the list
  - Deleting endpoint machines from the list
  - Viewing log information for the list and all associated machines
  - Changing delegations on the list

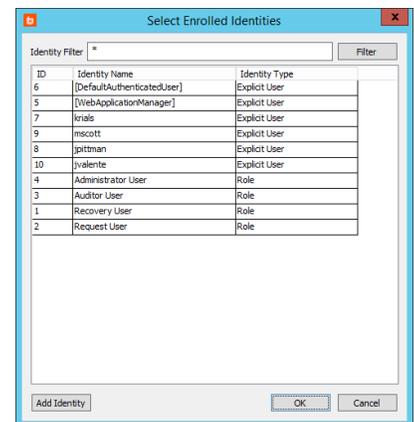
7. Check **Elevation Permissions** if this identity should be allowed to elevate user accounts on managed systems. Remember that checking this option allows the identity to elevate to all systems in the list.
8. Click **Create**.
9. After you've added an identity, you will see it in the delegation list. Click **Edit Policy** to modify its settings or **Delete Policy** to remove this identity's permissions to this list.

## Grant Permissions through the Management Console

1. In the management console, select **Delegation > Web Application Disconnected Account Permissions**.
2. Select a list to add a delegation to, then click **Add**.



3. From the **Select Enrolled Identities** dialog, choose one or more identities which should be allowed to access this list.



 **Note:** To add a new identity, click the **Add Identity** button, and complete the **Add Delegation Identity** information.

 For more information about adding identities, please see the [Privileged Identity Admin Guide \(PDF\)](http://www.beyondtrust.com/docs/privileged-identity/documents/pi-admin.pdf) at [www.beyondtrust.com/docs/privileged-identity/documents/pi-admin.pdf](http://www.beyondtrust.com/docs/privileged-identity/documents/pi-admin.pdf).

## Enroll Endpoints in a Disconnected Account List

To manage systems with disconnected account management (DAM), those systems must be enrolled in a DAM list. Endpoints can be enrolled as part of an online process, or they can be pre-enrolled. You can enroll disconnected systems either before or after you delegate permissions.

 For more information on delegating permissions, please see "[Delegate Permissions for Disconnected Account Access](#)" on page 14.

### IMPORTANT!

*The DAM client handles all management of enrolled systems and currently manages only one account per machine, specified at time of enrollment. If the DAM-managed account is "Administrator", then DAM manages only this account and no others. If Privileged Identity already has a password change job for this same account on the same system, then the Privileged Identity administrator must delete that job to prevent a password conflict.*

## Default Enrollment Process

From the disconnected account list, download and install the settings file and the client software on the endpoint.

 For more information about installing the endpoint client, please see "[Configure Endpoint Clients](#)" on page 19.

When running the client software, the first thing an endpoint does is attempt to communicate with the web service to update its registration with the target list. If the system has not been previously enrolled, it receives an identifier that is stored in the local settings and that is used to identify the endpoint for all future actions that will run against the web service. This endpoint ID is a GUID, so multiple machines with the same DNS name or virtual system settings will not be confused.

The next step of enrollment is to gather the endpoint's DNS name, IP address, MAC address, and platform information. This information is stored in the local settings and pushed to the server through the web service. This step is run each time the client software runs to ensure the system information has not changed. If a change is detected (for example, if locally stored settings don't match what the system reports), the information is updated on the server through the web service.

After establishing the machine ID and endpoint machine settings, the next step is to download the password and secret renewal policy. This determines how often the password will change and how often the endpoint must reconnect to the web service to obtain a new secret. The client software now begins the randomization process. It is not required to connect to the back-end web service until a new secret is required.

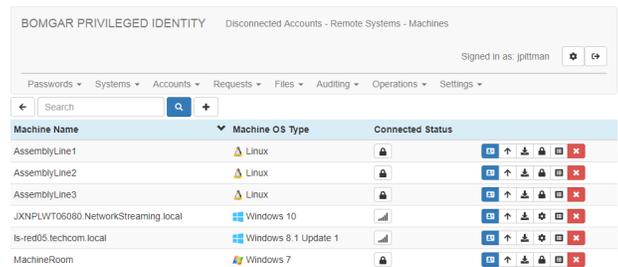
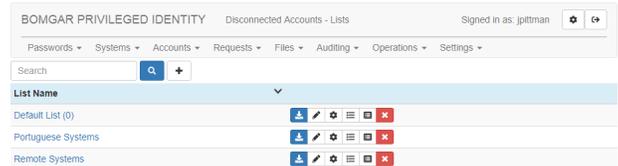
 **Note:** After initial enrollment, the endpoint is not strictly required to connect to the web service. As scheduled by its policy, the endpoint will attempt to reach the web service to update its status and get a new shared secret. If it can't reach the web service, the endpoint will continue to use the existing secret and settings to generate derived passwords.

*While password randomization will continue past the secret's expiration, the passwords on the web service will no longer match the passwords on the endpoint. This is not considered a failure, though the web app will warn you of the endpoint's expired secret. The endpoint will keep trying to reach the web server to update its secret, policy, and status.*

## Pre-Enrollment Process

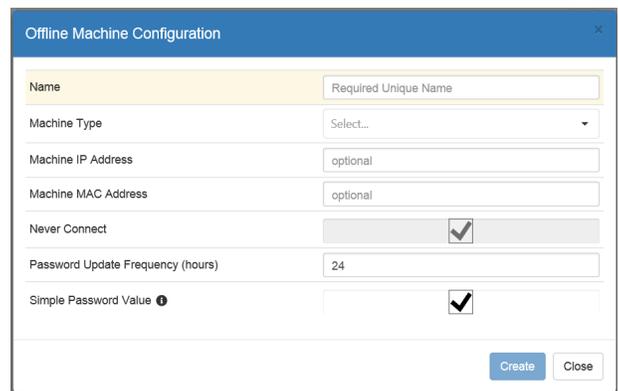
To pre-enroll an endpoint, you must add an entry for the endpoint to an existing list. Then, that endpoint's specific settings file is downloaded, and the client software and specific **settings.json** file must be distributed to and installed on the endpoint.

1. Log into the web application as a user with appropriate permissions.
2. Go to **Passwords > Disconnected Accounts**. You will see any lists you may access.
3. Click a list name to view its enrolled systems.
4. Click the **New Offline Machine** button (+).



5. Enter the following information:

- **Name:** Enter a descriptive name for the endpoint.
- **Machine Type:** Select the endpoint's operating system.
- **Machine IP Address:** (*Optional*) Enter the endpoint's IP address.
- **Machine MAC Address:** (*Optional*) Enter the endpoint's MAC address.
- **Never Connect:** Select this option if you know that this client will never connect to the web service to update its settings and status. This option poses no technical problems whether it is selected or not. However, selecting this option prevents misreporting of the client status in the web app.
- **Password Update Frequency:** Select how many hours to wait between password rotations.
- **Simple Password Value:** If you select this option, then a password derived from the client secret will be a 14-character, random string of uppercase letters, lowercase letters, and numbers.



If you do not select this option, then set the following options for passwords derived from the client secret:

- **Password Length:** Set how many characters to include in the password. The maximum is 127 characters.
- **Allow Numbers:** Set if the password can contain numbers.
- **Allow Lowercase:** Set if the password can contain lowercase letters.
- **Allow Symbols:** Set if the password can contain special characters.
- **Symbol Set:** You can leave the text field blank to allow all possible symbols, or you can define an allowed list of symbols.

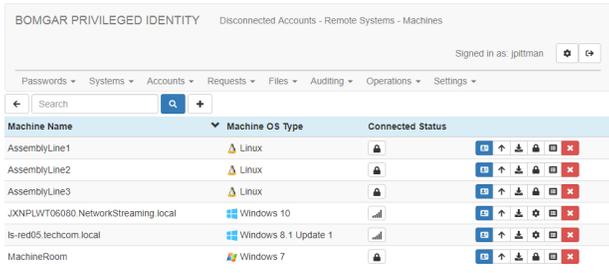


**Note:** To avoid causing code issues, you may not specify a slash (/), backslash (\), colon (:), semicolon (;), or quotation mark (").

 Some databases accept only the special characters hash (#), underscore (\_), and dollar sign (\$)

- Click **Create** to add the endpoint.
- You should now see the endpoint you just created. You can further edit its properties by clicking the buttons:

- **Show Password:** View and copy the endpoint's current password and its next password, as well as how much longer the current password will remain valid.
- **Create Elevation Code:** View, copy, or create an elevation code. You may also edit the elevation details, including the local group, the elevation duration, and the elevation start time.
- **Downloads:** Download the settings file and the client software - either the Windows service or the Python script. You must download the **settings.json** file to the directory where you will run the client software installer.
- **Edit Offline Configuration:** Modify the machine's name, type, IP address, MAC address, connection check, password update frequency, and password settings.
- **Show Logs:** View event logs for this machine.
- **Delete Machine:** Remove this machine from the disconnected account list.



- After you download and install the settings file and the client software, the client begins the randomization process. It is not required to connect to the back-end web service until a new secret is required.

 **Note:** After initial enrollment, the endpoint is not strictly required to connect to the web service. As scheduled by its policy, the endpoint will attempt to reach the web service to update its status and get a new shared secret. If it can't reach the web service, the endpoint will continue to use the existing secret and settings to generate derived passwords.

While password randomization will continue past the secret's expiration, the passwords on the web service will no longer match the passwords on the endpoint. This is not considered a failure, though the web app will warn you of the endpoint's expired secret. The endpoint will keep trying to reach the web server to update its secret, policy, and status.

## Configure Endpoint Clients

A newly enrolled endpoint does not need to know about password policies or settings; it must know only the list ID and the web service URI from which it should download its password secret and policies. Disconnected clients can be Windows, Mac, or Linux systems.

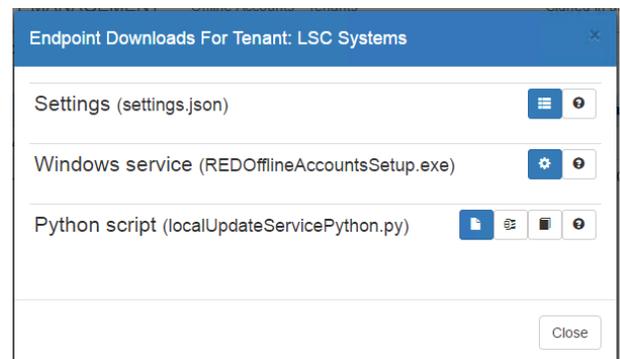
The endpoint client software is provided in two formats: a Windows service and a Python script. The Python script functions on both Windows and non-Windows systems. Both forms of the client software are available as downloads from the web application.



**Note:** Endpoints using the Windows service must have .NET framework 4.5.2 or later installed. Endpoints using the Python script must have anything in the Python 2.x family from 2.6.x and up. For Privileged Identity 7.3.0 and newer, you may use Python 2.6.x and up or Python 3.x.

### Download Installer Files

1. In the web application, select **Passwords > Disconnected Accounts**.
2. On the target list or endpoint, click the **Downloads** button.
3. Download the settings file and the client software - either the Windows service or the Python script. You must download the **settings.json** file to the directory where you will run the client software installer.



### Configure the Settings File

The settings file includes the web service URI and the list ID. It also contains information about offline account elevation. If you need to change any of these settings, such as if the web service is installed to a different location, modify **settings.json** using a text editor.

```
{
  "ServiceURI": "HTTP://SERVER.EXAMPLE.LOCAL:80/ERPWEBSEVICE/OfflineUpdateWebService.svc/",
  "TenantGUID": "ca388a55-8f26-87c8-8b54-720556d5ac74",
  "EnableOfflineElevation": true,
  "OfflineElevationWindow": 0,
  "OfflineForceLogout": false
}
```

By default, the endpoint updates the Windows or Mac administrator account or the Linux root account. If you want to manage an account other than the default, add the value **LocalAccountName** to the settings file:

```
{
  "LocalAccountName": "jsmith",
  "ServiceURI": "HTTP://SERVER.EXAMPLE.LOCAL:80/ERPWEBSEVICE/OfflineUpdateWebService.svc/",
}
```

```
"TenantGUID": "ca388a55-8f26-87c8-8b54-720556d5ac74",  
"EnableOfflineElevation": true,  
"OfflineElevationWindow": 0,  
"OfflineForceLogout": false  
}
```



**Note:** If you download the **settings.json** file for a specific machine, it will contain additional fields about endpoint machine identification, password policy settings, and the shared secret. If you download **settings.json** from the parent list, these additional fields are added the first time the client connects to the web service.

## Endpoint Settings and Local Secret Storage

After installation, the Python script stores its local settings in a **settings.json** file, while the Windows service stores its settings in the registry under **HKLM\Software\Wow6432Node\Lieberman\OfflineUpdateService**.

When the endpoint client runs, its status is saved in its local settings. These settings contain the last run time, the machine ID, the client software version, the result code of the last operation, and a status message. If the endpoint can reach the web service, this information is uploaded to the server and is visible in the logs.

The local settings also contain the **OfflineSecret**. If an endpoint machine does not have a stored secret, or if the secret has expired, the endpoint will obtain a new secret from the web service. The secret is generated on the server using a strong PRNG and is represented as a 32-character string. This secret is saved on the server and associated with the machine ID. It is then downloaded to the endpoint and saved in the local settings.

Because **OfflineSecret** is the only sensitive value that is locally stored, it is important to protect it. For Windows endpoints, this means running the service as **LocalSystem** and using **ProtectedData** to write the encrypted value of the secret to the registry. For Python endpoints, it means making the **settings.json** file accessible only to root for read, write, and execute (rwx).



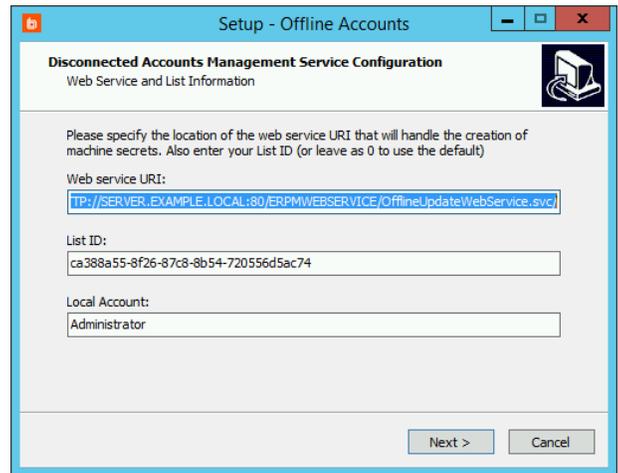
**Note:** After initial enrollment, the endpoint is not strictly required to connect to the web service. As scheduled by its policy, the endpoint will attempt to reach the web service to update its status and get a new shared secret. If it can't reach the web service, the endpoint will continue to use the existing secret and settings to generate derived passwords.

While password randomization will continue past the secret's expiration, the passwords on the web service will no longer match the passwords on the endpoint. This is not considered a failure, though the web app will warn you of the endpoint's expired secret. The endpoint will keep trying to reach the web server to update its secret, policy, and status.

## Install and Run the Windows Service

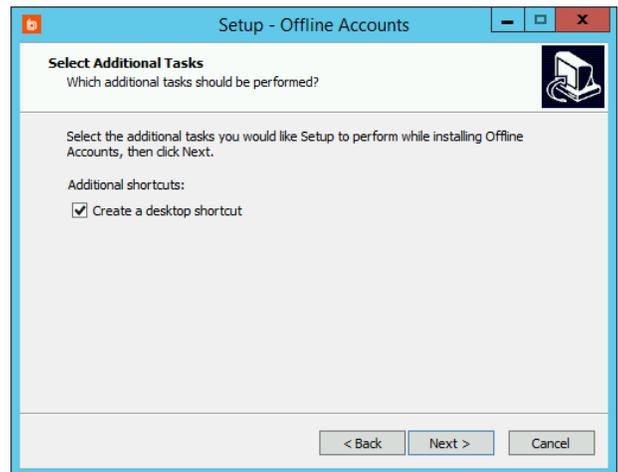
1. Log into the endpoint machine as an administrator.
2. Locate **REDOfflineAccountsSetup.exe**. Make sure that **settings.json** is in the same folder. Run the installer.

3. Verify and/or correct the following information, supplied from **settings.json**:
  - Web service URI
  - List ID
  - Local account
4. Click **Next**.

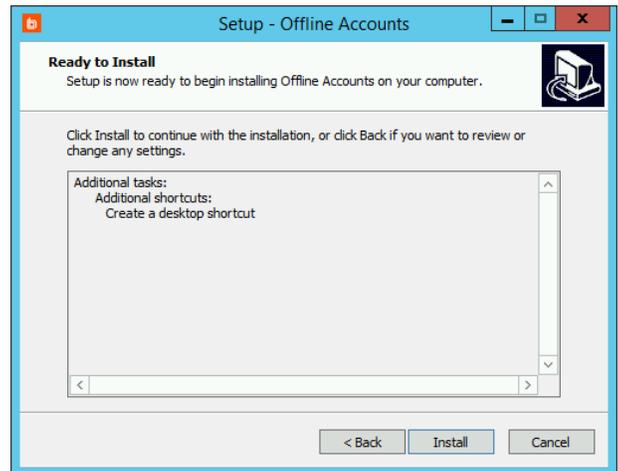


 **Note:** If you need to update the web service URI or the list ID later, you can reinstall the client or edit `ImagePath` in the registry at `HKEY_LOCAL_MACHINE\SYSTEM\ControlSet\Services\LiebssoftLocalUpdates`. Then restart the service.

5. Choose if you want to create a desktop shortcut, then click **Next**.



6. Click **Install**.



7. You can choose to start the **LiebssoftLocalUpdates** service after install.

8. Click **Finish**.



When the local service starts, it creates a **status.json** file in the same directory as that service. This file contains the machine ID, the last update time, the version, the status code, and the status message. This information is pushed to the web service and is available for view in the web app. The local service is set to run automatically on startup, so machines that are turned off will resynchronize immediately once turned on and once the web service is available.



**Note:** If the web service is unavailable when you first configure the local service, the local service starts but does not randomize any passwords. It continues to poll the web service every 60 minutes until it successfully contacts the web service and downloads its client settings. If the client has had at least one successful connection to the web service, then in the future, the local service will randomize passwords even when the web service is unavailable.

## Manage Client Logging

The local service log file is located under the installation directory. The default is **C:\Program Files (x86)\Lieberman\Offline Accounts\Logs**.

The endpoint software attempts to connect to the web service to update the server with log messages as actions occur. If the web service is unavailable while the endpoint client is running, the log information is not sent to the server. Logs that are successfully sent are accessible through the web app either at the list or machine level.

To control logging, open the registry to **HKEY\_LOCAL\_MACHINE\SOFTWARE\Wow6432Node\Lieberman\OfflineUpdateService**. Modify **LoggingThreshold** to:

- **0:** Trace. **Warning: This log includes the password being set.**
- **1:** Verbose. Logs errors, successes, and detailed messages.

- **2:** Logs success and failure messages.
- **3:** Logs error messages only.

## Launch the Local Password Client

Open the **Offline Accounts** software. This requires admin rights on the host system.



**Note:** If the local service has not connected to the web service at least once, several fields in the console will be blank.

The local password client displays information about the host machine, password changes, password policies, and the local service.

- **Service URI:** The URI of the web service.
- **Machine ID:** The unique ID used to distinguish this endpoint.
- **Machine DNS name:** This endpoint's DNS name.
- **Machine IP address:** This endpoint's IP address.
- **Machine MAC address:** This endpoint's MAC address.
- **Local account name:** The account managed by the disconnected account software.
- **Machine type:** The details of this endpoint's OS version.
- **Secret set time:** The time when the client secret was last set.
- **Next secret update:** The time when the client secret is scheduled to update.
- **Password set time:** The time when the service last set the password.
- **Next password update:** The time when the service will update the password.
- **Password changes:** The number of times the service has reset the password.
- **Password Policy:** The password requirements as defined on the web service.



The screenshot shows the 'Local Password Client' window. It has a light blue title bar and a white background. The fields are as follows:

- Service URI: HTTP://SERVER.EXAMPLE.LOCAL:80/ERPMBWESERVICE/OfflineUpdate/WebService.svc/
- Machine ID: {79F81C81-A974-4532-8AEE-B8E2B330}
- Machine DNS name: ls-red05.techcom.local
- Machine IP address: 10.102.10.82
- Machine MAC address: 00:50:56:88:CC:C3
- Local account name: Administrator
- Machine type: Microsoft Windows NT 6.3.9600.0
- Secret set time: 9/25/2018 4:16:20 PM
- Next secret update: 9/25/2018 4:16:20 PM
- Password set time: Not Set
- Next password update: Not Set
- Password changes: 0
- Password Policy:
  - Use simple hash
  - Password length: 14
  - Allow numbers
  - Allow lowercase
  - Allow symbols
- Local service status: Running
- Client version: 1700531

Buttons at the bottom: Stop Service, Start Service, Update Secret, Test Connection to Web Service.



For more information about password policies, please see "[Manage Password Policies for Disconnected Accounts](#)" on [page 9](#).

- **Local service status:** The current state of the local service.
- **Client version:** The software version number.
- **Stop Service:** Stop the local service.
- **Start Service:** Start the local service.
- **Update secret:** Connect to the web service to force an update of the shared secret.
- **Test Connection to Web Service:** Check the connection to the web service.

## Run Client Commands

The client application has several command line arguments that can be run to help with diagnostics. Run the following commands from an administrative command prompt:

- `LocalPasswordClient.exe WriteStatus` - Returns the most recent status of the local service.
- `LocalPasswordClient.exe TestSetPassword NewPassword` - Runs the code the local service uses to set the password for a local account. This command can be used to diagnose problems if the service fails to update the password. Make sure you run this command in the same context as the local service (**LocalSystem**) to replicate the service behavior.
- `LocalPasswordClient.exe IterationTest` - Calculates and times 10,000 derived password generations. This can help determine if the password length or character set overloads the system when deriving passwords. Make sure you run this command in the same context as the local service (**LocalSystem**) to replicate the service behavior.
- `LocalPasswordClient.exe Secret` - Returns the currently stored secret. Make sure you run this command in the same context as the local service (**LocalSystem**) to replicate the service behavior.
- `LocalPasswordClient.exe GeneratePassword Secret 10 14 1 1 1 0` - Generates a password derived from the provided secret and with the provided settings. Input arguments are, in order:
  - **Secret:** (string) 32-character string.
  - **Iteration Count:** (int) The number of iterations of derived passwords to calculate.
  - **Length:** (int) The derived password length.
  - **Numbers:** (int) Allow numbers in the derived password.
  - **Symbols:** (int) Allow symbols in the derived password.
  - **Lowercase:** (int) Allow lowercase letters in the derived password.
  - **Simple Hash:** (int) Use simple hash password generation. If set to **1**, the length, numbers, symbols, and lowercase settings are ignored, though they still must be provided.

Make sure you run this command in the same context as the local service (**LocalSystem**) to replicate the service behavior.

- `LocalPasswordClient.exe PasswordTest` - Returns the current derived password based on the stored settings. Make sure you run this command in the same context as the local service (**LocalSystem**) to replicate the service behavior.
- `LocalPasswordClient.exe Password` - Returns the current derived password and secret based on the stored settings. Make sure you run this command in the same context as the local service (**LocalSystem**) to replicate the service behavior.



**Note:** To execute a command as *LocalSystem* if you have administrative permissions, you can impersonate *LocalSystem* or use an application like *PsExec* to do so. For example:

```
psexec -s "C:\Program Files (x86)\Lieberman\Offline Accounts\LocalPasswordClient.exe"
password
```

The output would be similar to:

```
Current password:
jcr!Cl*{#;}^#l0~hr Z
Current secret:
kRqsrwwovoJqCgzt87Ji7mSNAqhKKAfY
```

**i** For more information about PsExec, please see [docs.microsoft.com/en-us/sysinternals/downloads/psexec](https://docs.microsoft.com/en-us/sysinternals/downloads/psexec).

**i** For more information about the LocalSystem context, please see "Endpoint Settings and Local Secret Storage" on page 20.

## Windows Client Settings

The Windows endpoint client can accept settings from multiple sources. You can specify settings through one or more of these input methods. Settings from multiple sources are applied in the following order:

- Settings passed on the command line to the local service at startup. These settings must be specified as part of the service configuration during installation.
- Settings passed as part of the **OfflineUpdateService.exe.config** file. This file is located in the same directory as the local service executable.
- Settings saved to `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Lieberman\OfflineUpdateService` in the local registry. While settings stored in this key preserve the state of the disconnected service itself, you can overwrite or add to these settings either on startup or as it is running.

## Install and Run the Python Script

1. Log into the endpoint machine as an administrator.
2. Locate the installer package. Make sure that **settings.json** is in the same folder.
3. If you downloaded the **.tar** or **.zip** version of the installer, extract the file.
4. You may move **localUpdateServicePython.py** and **settings.json** to another location. Make sure you keep the two files together, and make sure you make note of the path.



**Note:** For a Linux distribution, we recommend that you save the Python script and settings files to a path accessible only to root, such as `/root/bin/REDOA` or `/bin/REDOA`.

## Run the Python Script

The script must be run as root or equivalent. Set up a `cron` job or `contab` job to schedule the script to run as often as needed. You can use `crontab -e` or `sudo contab -e` to edit the contab file as root. Add a line to the file to create the new job. For example, if you want the script to run every minute, the new line would look like this:

```
*/* * * * * python /root/bin/REDOA/localUpdateServicePython.py > /root/bin/REDOA/cron.log
```

In the example above, the path to Python is set in the environment. If Python is not part of the path or if the system must use a different version of Python than the one specified in the path, you can use the full path to the Python distribution, such as `/etc/python27/python`.

If you don't provide a **settings.json** file to the execution directory, you can specify the list ID and web service URI as command line parameters to the Python script:

```
*/* * * * * python /root/bin/REDOA/localUpdateServicePython.py 0 https://server.example.int/erpmwebservice/OfflineUpdateWebService.svc/ > /root/bin/REDOA/cron.log
```



**Note:** If you specify command line arguments as input to the script, those settings are used instead of the settings specified in the **settings.json** file, even if the file is present.

The first part of the contab entry is the scheduling frequency. Next are the path to Python and the path to the installer. You may also include the list ID, the web service URI, the path to the cron file for logging job results, the service certificate file path, and the local account name.



### IMPORTANT!

The trailing slash on the web service URI is required.

## Python Script Options

When executed from the command line, the Python script includes several command line arguments intended to help diagnose health and status of the endpoint. The command line arguments are single strings. You should pass only one argument at a time. For example:

```
$ python /root/bin/REDOA/localUpdateServicePython.py SecretAndPassword
```

Available commands:

- `Version` - Returns the endpoint client version.
- `SecretAndPassword` - Returns the currently stored secret, the number of times a password has been generated since the secret was created, and the current derived password.
- `LastPassword` - Returns the last password based on the recorded last password set time.
- `Settings` - Returns the current endpoint client settings.
- `Reset` - Clears all locally saved settings except the web service URI and list ID. This causes the endpoint client to get a new machine ID, to request a new secret, and to trigger an immediate password change the next time the script is run normally.

## Client Logging

The service log is located in the same directory as the Python script and is named **localupdateservicepython.log**. The Python script includes flags that control the logging threshold (verbosity and log-to-console output) and the ability to clear the log file before each run.

The endpoint software attempts to connect to the web service to update the server with log messages as actions occur. If the web service is unavailable while the endpoint client is running, the log information is not sent to the server. Logs that are successfully sent are accessible through the web app either at the list or machine level.

# Access Disconnected Passwords and Elevate Disconnected Accounts

You can access disconnected passwords or elevate a disconnected account from the web application, the web service, PowerShell, or the client directly. In this section, we describe the methods used and the configurations required for disconnected account management (DAM).

**i** To retrieve a disconnected password or to elevate a disconnected account, you must use an all-access account or an account that has been delegated permissions to the list. For more information, please see *"Delegate Permissions for Disconnected Account Access"* on page 14.

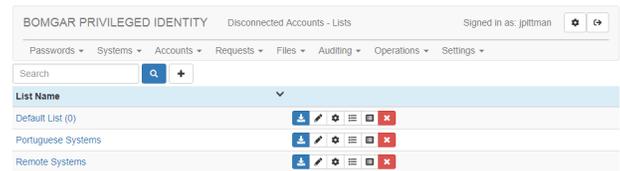
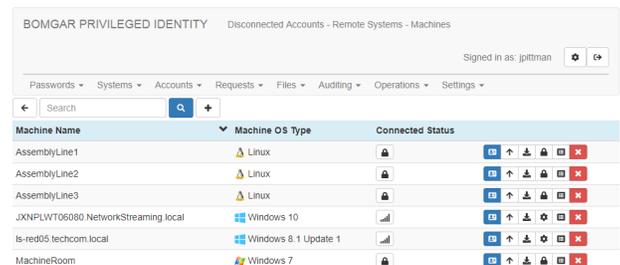
## Retrieve a Disconnected Password from the Web Application

1. Log into the web application as a user with appropriate permissions.
2. Go to **Passwords > Disconnected Accounts**. You will see any lists you may access.
3. Click a list name to view its enrolled systems.
4. Click the **Show Password** button (blue badge) to view the current and next passwords.

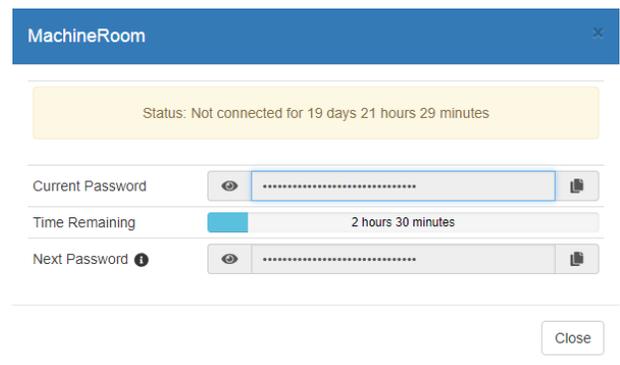
**i** To elevate a disconnected account instead, please see *"Elevate a Disconnected Account"* on page 28.

5. The web app shows the current derived value of the password based on the stored secret, the time that's passed since the secret was generated, and the number of iterations that have passed since that time and the current time. The password display also shows the next expected password.

You can also see the endpoint status. If the endpoint synchronized with the web service at the most recent expected time, the status is **Online**. If it did not synchronize at that time, the status shows how long it has been disconnected. If the endpoint has never synchronized with the web service, the status is **Always Offline**.

Machine Name	Machine OS Type	Connected Status
AssemblyLine1	Linux	Online
AssemblyLine2	Linux	Online
AssemblyLine3	Linux	Online
JXNPLWT06080.NetworkStreaming.local	Windows 10	Offline
Is-red05.techcom.local	Windows 8.1 Update 1	Offline
MachineRoom	Windows 7	Offline



## Retrieve a Disconnected Password with the Windows Service

From an administrative command prompt, run `LocalPasswordClient.exe Password`. This returns the current derived password and secret based on the stored settings. Make sure you run this command in the same context as the local service (**LocalSystem**) to replicate

the service behavior.



**Note:** To execute a command as LocalSystem if you have administrative permissions, you can impersonate LocalSystem or use an application like PsExec to do so. For example:

```
psexec -s "C:\Program Files (x86)\Lieberman\Offline Accounts\LocalPasswordClient.exe"
password
```

The output would be similar to:

```
Current password:
jcr!Cl*{#;}^#l0~hr Z
Current secret:
kRqsrwwovoJqCgzt87Ji7mSNAqhKKAfy
```



For more information about PsExec, please see [docs.microsoft.com/en-us/sysinternals/downloads/psexec](https://docs.microsoft.com/en-us/sysinternals/downloads/psexec).



For more information about the LocalSystem context, please see "Endpoint Settings and Local Secret Storage" on page 20.



For more information about Windows commands, please see "Run Client Commands" on page 24.

## Retrieve a Disconnected Password with Python

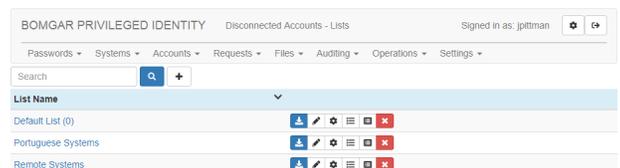
From the command line, run: `$ python /root/bin/REDOA/localUpdateServicePython.py SecretAndPassword`. You must run this command with sufficient privileges to access the `settings.json` file (root by default).



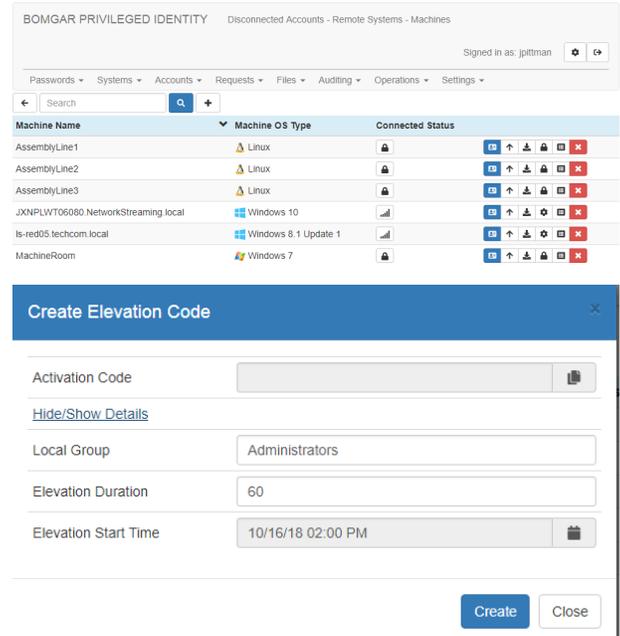
For more information about Python commands, please see "Python Script Options" on page 26.

## Elevate a Disconnected Account

1. Log into the web application as a user with appropriate permissions.
2. Go to **Passwords > Disconnected Accounts**. You will see any lists you may access.



3. Click a list name to view its enrolled systems.
4. Click the **Create Elevation Code** button (up arrow) to get a code to elevate an account on the desired endpoint.
5. When the dialog appears, you can click **Create** to immediately create an elevation code with the default settings. To change the settings, click **Hide/Show Details**.
  - **Local Group:** Enter the name of the group to which you wish to elevate the account.
  - **Elevation Duration:** Set how many minutes the elevation should last.
  - **Elevation Start Time:** Set the date and time for the elevation code to become active.
6. Click the **Copy** button to copy the **Activation Code** to your clipboard.



The screenshot shows the BOMGAR PRIVILEGED IDENTITY management console. At the top, it displays 'Disconnected Accounts - Remote Systems - Machines' and the user 'jptlman'. Below this is a navigation menu with options like Passwords, Systems, Accounts, Requests, Files, Auditing, Operations, and Settings. A search bar is present. The main area contains a table of machines:

Machine Name	Machine OS Type	Connected Status
AssemblyLine1	Linux	Connected
AssemblyLine2	Linux	Connected
AssemblyLine3	Linux	Connected
JXNPLWT06080.NetworkStreaming.local	Windows 10	Connected
Is-red05.techcom.local	Windows 8.1 Update 1	Connected
MachineRoom	Windows 7	Connected

Below the table is a 'Create Elevation Code' dialog box with the following fields:

- Activation Code: [Text field with copy icon]
- Local Group: Administrators
- Elevation Duration: 60
- Elevation Start Time: 10/16/18 02:00 PM

Buttons for 'Create' and 'Close' are at the bottom right of the dialog.

 **Note:** If you select **Today** from the calendar, it automatically selects the current hour. If you want a different time, pick a date from the calendar.

 **IMPORTANT!**

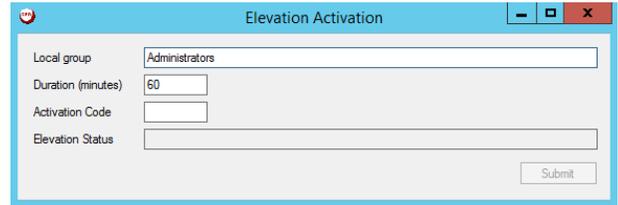
Make sure you remember any modifications you make to the default settings, as this information must match the information in the endpoint client. There are five checks: activation code, local group, elevation duration, time stamp, and shared secret. If any of these data don't match, the elevation will not succeed.

 **Note:** This code remains valid for the length of time set in the management console, plus the number of minutes remaining until the top of the hour. For example, if the settings dictate that codes should last for one hour and if you check out a code at 8:15, the code will remain valid until 10:00. You can reuse that same code any number of times within this time frame. If you need to elevate the account after this code has expired, you must go through the account elevation process again.

This code window is not related to the elevation duration. As long as the elevation begins while the code is valid, the elevation can continue past the code expiration.

 For more information, please see "[Disconnected Elevation Settings](#)" on page 6.

7. On the disconnected endpoint, open **LocalElevationClient.exe**, installed by default at **C:\Program Files (x86)\Lieberman\Offline Accounts\LocalElevationClient.exe**, and typically accessible through the desktop shortcut **Lieberman RED Offline Accounts Elevation**.
8. Enter the activation code and modify any settings you changed in the web app. Then click **Submit**. The account elevates to the specified group for the set length of time.



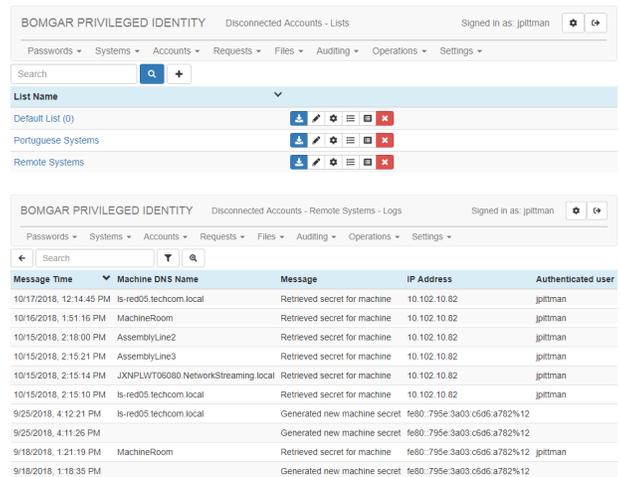
**Note:** After an account has been de-elevated, a setting in the management console can force the account to be logged off the managed system.

# Review Logs for Disconnected Account Activity

Disconnected account management (DAM) captures two sets of logs. These are separate from the other Privileged Identity logs. Users with all-access permissions and delegated users with write permissions to a list can view DAM logs.

## View List Logs from the Web Application

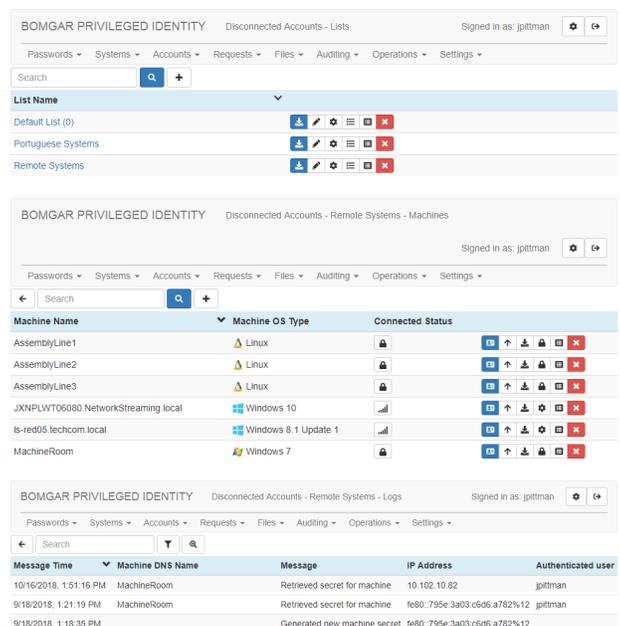
1. Log into the web application as a user with appropriate permissions.
2. Go to **Passwords > Disconnected Accounts**. You will see any lists you may access.
3. Click the **List Logs** button for the list you want to view.
4. You can find log entries using the search box and filter button at the top of the list. You can also view more detailed logs.



Message Time	Machine DNS Name	Message	IP Address	Authenticated user
10/17/2018, 12:14:45 PM	Is-red05.techcom.local	Retrieved secret for machine	10.102.10.82	jpitman
10/16/2018, 1:51:16 PM	MachineRoom	Retrieved secret for machine	10.102.10.82	jpitman
10/15/2018, 2:18:00 PM	AssemblyLine2	Retrieved secret for machine	10.102.10.82	jpitman
10/15/2018, 2:15:21 PM	AssemblyLine3	Retrieved secret for machine	10.102.10.82	jpitman
10/15/2018, 2:15:14 PM	JXNPLWT06080.NetworkStreaming.local	Retrieved secret for machine	10.102.10.82	jpitman
10/15/2018, 2:15:10 PM	Is-red05.techcom.local	Retrieved secret for machine	10.102.10.82	jpitman
9/25/2018, 4:12:21 PM	Is-red05.techcom.local	Generated new machine secret	fe80:795e:3a03:c6d6:a782%12	
9/25/2018, 4:11:26 PM		Generated new machine secret	fe80:795e:3a03:c6d6:a782%12	
9/19/2018, 1:21:19 PM	MachineRoom	Retrieved secret for machine	fe80:795e:3a03:c6d6:a782%12	jpitman
9/19/2018, 1:18:35 PM		Generated new machine secret	fe80:795e:3a03:c6d6:a782%12	

## View System Logs from the Web Application

1. Log into the web application as a user with appropriate permissions.
2. Go to **Passwords > Disconnected Accounts**. You will see any lists you may access.
3. Click a list name to view its enrolled systems.
4. Click the **List Logs** button for the system you want to view.
5. You can find log entries using the search box and filter button at the top of the list. You can also view more detailed logs.



Machine Name	Machine OS Type	Connected Status
AssemblyLine1	Linux	Connected
AssemblyLine2	Linux	Connected
AssemblyLine3	Linux	Connected
JXNPLWT06080.NetworkStreaming.local	Windows 10	Connected
Is-red05.techcom.local	Windows 8.1 Update 1	Connected
MachineRoom	Windows 7	Connected

Message Time	Machine DNS Name	Message	IP Address	Authenticated user
10/16/2018, 1:51:16 PM	MachineRoom	Retrieved secret for machine	10.102.10.82	jpitman
9/19/2018, 1:21:19 PM	MachineRoom	Retrieved secret for machine	fe80:795e:3a03:c6d6:a782%12	jpitman
9/19/2018, 1:18:35 PM		Generated new machine secret	fe80:795e:3a03:c6d6:a782%12	

# Remove Disconnected Account Lists and Clients

As machines are removed from the network or as lists are no longer serviced, you will need to remove their data from Privileged Identity. In this section, we discuss the removal and cleanup process.

## Remove Clients from Lists

A user with all-access permissions can remove a client from any list. Additionally, a delegated user with write permission to a list can remove a client from their permitted lists.

Removing a client from the list deletes all server-side information associated with that endpoint, including:

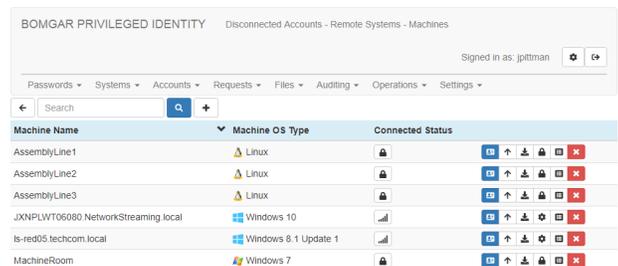
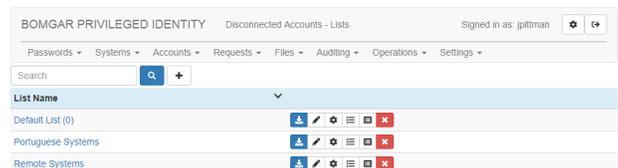
- Machine information (based on machine ID)
- Machine secret
- Policy information for that machine
- Log information for that machine

### IMPORTANT!

*If the endpoint software is still running, the client is recreated and added to the list the next time the endpoint synchronizes with the server. If you intend to remove a client permanently, you must first remove the endpoint software from the machine.*

## Remove a Client through the Web Application

1. Log into the web application as a user with appropriate permissions.
2. Go to **Passwords > Disconnected Accounts**. You will see any lists you may access.
3. Click a list name to view its enrolled systems.
4. Click the **Delete** button (red **X**) for each client you wish to remove.
5. You will be prompted to confirm that you want to delete the client.



## Remove Lists

A user with all-access permissions can remove a list entirely. Deleting a list deletes all server-side information associated with the list, including:

- List information (based on list ID)
- Machines associated with the list
  - Machine information (based on machine ID)
  - Machine secret
- Password policy information
- List permissions
- All database logs for the list



**Note:** Text log files created by the server are not deleted as part of this operation.

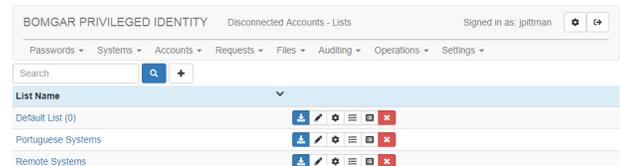


## IMPORTANT!

*If the endpoint software is still running on machines associated with the deleted list, and if the server is configured to allow endpoints to automatically create new lists, then the list will be recreated the next time as associated endpoint synchronizes with the server. However, existing permissions and logs as well as all other endpoint registrations will not be recreated.*

## Remove a List from the Web Application

1. Log into the web application as a user with appropriate permissions.
2. Go to **Passwords > Disconnected Accounts**.
3. Click the **Delete** button (red **X**) for each list you wish to remove.
4. You will be prompted to confirm that you want to delete the list.



# Notes about Disconnected Account Management

## Duplicate System Resolution

If multiple endpoints are enrolled in the same list and both report the same non-blank DNS name, IP address, and MAC address, the machine ID information stored on the server will be merged into a single entry. This happens by removing the data for all but the most recently created machine ID. If both endpoints are valid systems, they will continue to change the local password and update the offline secret, but the duplicate entries will not be shown in the web application.

## OS X Endpoints

Apple OS X has a root account that by default does not have a password. The single-user installation case makes the user created at setup able to access all root permissions using `sudo`. Users on OS X are typically managed using the `dsccl` command to target users in the `/Users` directory.

Because there is no standard default user on OS X, the **LocalAccountName** parameter should be set in **settings.json** to target a specific user on that endpoint machine.

The Python script can still be run as **root** on OS X by installing the script using `sudo`. This prevents other users from being able to read/write/execute the files without root permissions.

## Python 2.6.x

While Python 2.6.x is supported on the endpoint, SSL security context was not implemented in this version of Python. Therefore, if SSL is enabled on the web service host, its SSL certificate will not be validated by a client endpoint using Python 2.6. Also, if the web service requires SSL functionality such as client certificate validation or Integrated Windows Authentication, the endpoint will fail. We recommend using the most current version of Python to avoid these errors.

## Additional Resources

- **.NET ProtectedData**

[https://msdn.microsoft.com/en-us/library/system.security.cryptography.protecteddata\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.security.cryptography.protecteddata(v=vs.110).aspx)

- **PsExec**

<https://technet.microsoft.com/en-us/sysinternals/bb897553.aspx>