BOMGAR™

Privileged Identity
Disconnected Account Management

TC:5/21/2018

Table of Contents

Privileged Identity Disconnected Account Management

Privileged Identity is a solution designed to establish a base of knowledge regarding the systems and devices in your network, what accounts are on those systems and devices, and enable the 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 of systems difficult at best.

The patent pending Disconnected Account Management feature enables support for management of accounts (through changing passwords) that exist on disconnected machines.

The system works by generating cryptographically secure secret data on the server and allowing each client endpoint to pull data from the server to generate passwords derived from the shared secret to use for local accounts. Both the clients and server implement the same one-way hashing algorithms to derive a series of passwords from the stored secret data. Because the endpoints download the password policy settings from the server and thus share the same settings, both the endpoints and the server can at any time calculate the derived current password from the known secret.

Background and Goals

The Need for Strong Local Credentials

Organizations with a need for the most basic access security should use unique local logon credentials customized for each workstation and server in their environment. Unfortunately, most organizations use common credentials (same user name and password for the built-in administrator account) for each system for the ease of creating and managing those systems by the IT Department without any concern as to the consequences to the organization should these common credentials be compromised.

With the mandates of PCI-DSS, Sarbanes-Oxley, HIPAA, Gramm-Leach-Bliley, California Security Breach Information Acts, NASD 3010, SEC 17a-4, 21 CFR Part 11, DoD 5015.2 and others, the implementation of reasonably hard to compromise local logon credentials is mandatory for most organizations as a means for protecting not only the confidentiality of their data, but also to protect against tampering.

Creating Strong Local Credentials

Privileged Identity can change any common account on all workstations and servers in just a few minutes without the need for scripts or any other type of program. The new common credentials can be stored in a local or remote SQL Server database and can be recovered on demand using the web application.

Privileged Identity can be configured to regularly change the passwords of common accounts on all target systems (i.e. workstation built-in administrator account) according to a schedule so that each account receives a fresh cryptographically strong password regularly. This product feature protects the overall security of an organization so that the compromise of a single machine's local administrator password does not lead to the total compromise of the entire organization's security. Privileged Identity further builds on these concepts by automatically discovering all references to the specified account, such as services, tasks, COM and DCOM objects, and more, and following a password change for a user's account, whether domain or local, propagating the new password to all those references.

Delegated Password Recovery

Privileged Identity also contains a web client to allow the remote recovery of passwords, access to privileges sessions, and more. The web application is comprised of ASP and ASP.NET web pages that allows any user with the appropriate group memberships the right to use the application, as well as the right to recover passwords for accounts managed by the program. All access to the web application and all actions taken therein are logged, and the history is also available via the same web interface to authorized users.

Because this application protects and provides extremely sensitive information, it is essential that particular attention be payed to the security settings of the application and also use appropriate encryption such as SSL based on the scope of access provided.

Requirements and Setup

The Disconnected Account Management interface is a licensed feature of Privileged Identity and once licensed, must be enabled in the web application settings before any offline account updates can occur. This can be enabled directly in the web interface or in the web application properties via the management console. This setup will define the initial policy. Further policies can be defined for each disconnected systems list.

The Disconnected Account Management feature is supplied as a Windows service and as a Python application. The Python application will function on both Windows and non-Windows operating systems.

Clients will download and install the client software. The machines under management need to be connected to a network as part of the initial enrollment process. The enrollment process involves being able to connect and download policy information from a web service endpoint.

Server Configuration

An instance of Privileged Identity must be installed and configured in order to hold the protected machine secret information for the endpoints, as well as provide a back-end for the web service that will distribute the machine secrets. The disconnected password recovery portal can be managed via the web interface or web service.

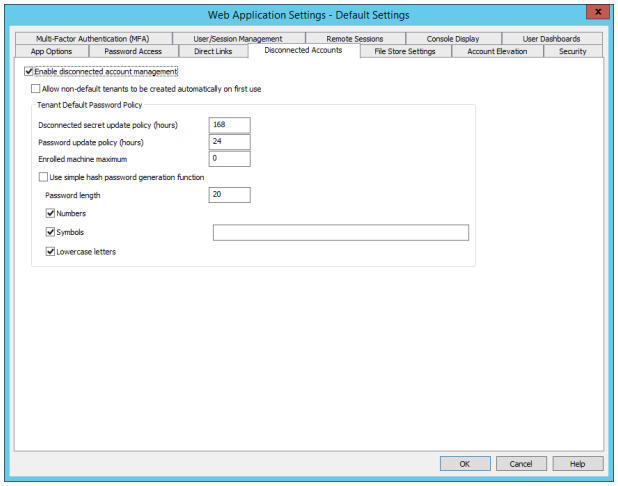
The disconnected password management endpoint web service is installed as part of the web service installation. See the Installation Guide for more information. After installation, the default disconnected account web service endpoint is accessible at:

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

IMPORTANT!

The web service endpoint must be enabled for anonymous authentication. If the web service is configured for integrated authentication or certificate authentication, the disconnected account management feature will not work. 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 the disconnected account management component of Privileged Identity, enable the feature in the web application settings property pages and supply default settings for generated password policy.



- **Enable disconnected account management** - Enables the disconnected account management feature.
- **Allow non-default tenants to be created automatically on first use** - Allows new tenants to automatically enroll. If a tenant ID does not exist and this option is enabled, when a new client checks in with a new tenant ID, a new tenant will be create with that ID.
- **Disconnected secret update policy (in hours)** - The frequency of the password change seed value change.
- **Password update policy (in hours)** - The frequency of the password change.
- **Enrolled machine maximum** - Maximum number of machines that a tenant can create.
- **Use simple hash password generation function** - When enabled, The first character will become a question mark and value containing uppercase, lower case, and numbers will be created. The length will be 31 characters. This password will be a derived value from the client secret.
- The following options are available when **Use simple hash password generation function** is not enabled:
    - **Password Length** - The length of the password to set.
    - **Numbers** - Allows numbers to be used in the password.
    - **Symbols** - Allows symbols to be used in the password. Leave the corresponding field empty to use all possible symbols or define an allowed list of symbols.
    - **Lowercase letters** - Allows lower case letters to be used in the password.

Disconnected Systems List Creation

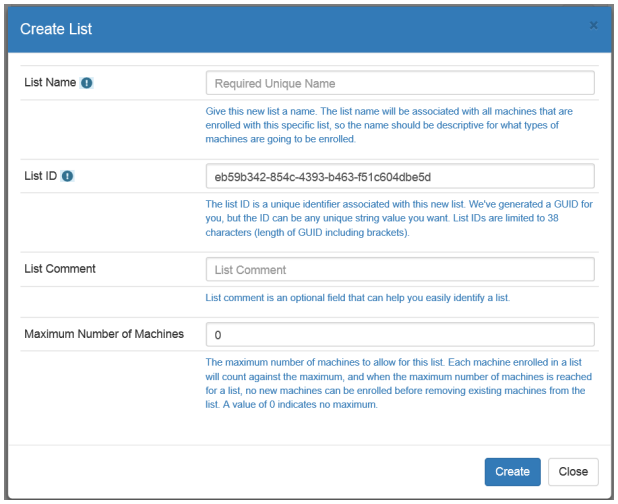
Lists of disconnected systems can be created automatically as part of new endpoint enrollment, or they can be created directly through the web application or web service.

If the web application setting is enabled to **Allow non-default lists to be created automatically on first use** and a client connects with a new List ID that has not previously been enrolled, a new List will be created and that system will be added to it.

Create a List via the Web Application

1. Login to the web application with all access permissions.
2. Go to **Passwords | Disconnected Accounts**.
3. Click **Add a Row (+)** at the top of the page.
4. Supply the following information to the **Create List** dialog:
    - **List name** - a friendly name for the List.
    - **List ID** - a List ID will be automatically generated GUID, however, you may use any string up to 255 characters if you prefer more human-readable information.
    - **List Comment - optional** - supply a comment or note for the list.
    - **Maximum Number of Machines** - the maximum number of allowable endpoints that can be added to the list. A value of 0 indicates an unlimited number of clients.
5. Click **Create**.

List settings may be edited later by clicking the **Edit** button at the end of the list's row.



IMPORTANT!

If you edit a List's ListID and save the changes, the old list with the old ID will still exist and a new list with the same name and the NEW ID will be created.

Create a List via the Programmatically

See the Programmers Guide for more information.

- There is no PowerShell cmdlet for adding or managing a list.
- From SOAP, call **EditTenant**.
- From REST, call **/REST/OfflineUpdate/Tenant**.

Password Policies

Derived Password Generation

Once a secret has been generated and stored on the endpoint, passwords can be derived. If a password set time has not been saved (no password has set/first run), or if the password expiration time has elapsed, a new password will be derived from the stored secret and set for the local account (root/administrator/other). The password expiration time is calculated based on the elapsed time from when the secret was last obtained from the server and the interval of password age defined in the password policy (in hours). Because the secret set time will always be known and shared by the endpoint and the server (because it is generated on the server and saved on the client) the set times on the endpoint cannot drift over extended operation. This means that even if the update process cannot run for an extended period of time, such as when an endpoint is offline/disconnected or suspended, the password will be changed immediately when update process is re-enabled (power-on or resumed) and the password will be re-synchronized with the value the value derived on the server. The endpoint also attempts to update the server with a most recent status each time a new secret is obtained or when a new password is derived and set, but a failure to update the server is expected in the disconnected case.

The algorithm used to derive the password from the stored secret depends on the password policy settings.

Two types of password policies can be created for default settings or a specific list:

- Simple Hashing Password Generation
- Admin defined Password Generation

Password policies allow you to configure the machine secret and password generation interval, the wild card matching string to determine which endpoints are affected by each policy, and define the format for the derived passwords used for the accounts on the endpoints. Password policies are copied to the endpoint as part of the endpoint creation process, and updated when the server generates a new secret for the endpoint (in case the policy has changed or a new policy applies to the endpoint machine).

By default, the password policy used for all endpoints will be defined on the server as part of the web application's settings. If not set, those settings will default to use the simple password hash method to derive passwords.

Endpoints will use exactly one policy.

If an endpoint matches more than one policy, the first matching policy will be applied. Endpoint machines will update the password policy they are using each time a new secret is generated by the server.

A password policy also defines how often the endpoint secret should be updated. This value is stored on the endpoint, and the endpoint will request a new secret be generated (and a new policy applied), when the time is reached. The time is measured as a differential between when the existing secret was generated and the current time on the endpoint (in hours). A password policy defines how often a new password should be derived from the stored secret and applied to the local account. This time is measured as a differential between the last time the password was set (in hours) and the current time. The password is always set immediately when a new secret is generated and stored on an endpoint. Password policies also define what the password format for the derived password will be, and what method of password generation for the derived password will be used.

Create a List Policy

To create a list policy, a list must first be created. See "Disconnected Systems List Creation" on page 7 for more information on creating a list.

List policies can be created 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. Click the **List Policies** button (gear icon) next to the list.
3. Click the **Add a Row** button (+) at the top of the page. The default policy elements will be filled in and can be edited.
4. Define the following policy elements:

   - **Machine Type Filter** - The default machine type filter is set to * which means it will apply to all systems enrolled in the list. See "Password Generation Policy Filter" on page 10 for more information on machine filters.
   - **Secret Age** - The default is 168 hours. The secret is essentially a seed value for the password generator.
   - **Password Age** - The default is 24 hours. The target account's password will change every 24 hours.
   - **Use Simple Password** - Enable this option to enable the Simple Hashing Password Generation policy. Do not enable this option to defined an actual password policy. See "Password Generation Settings" on page 12 for more information.
5. Click the **Save** button.

Create a List Policy Programmatically

See the programmers guide for more information.

- There is no equivalent command for PowerShell at this time.
- From SOAP, call **EditPasswordPolicy**.
- From REST, call **/REST/OfflineUpdate/PasswordPolicy**.

Password Generation Policy Filter

When enabling the Disconnected Account Management feature, a default password policy was defined. Password policies can also be defined on a per list basis. If no list specific policy has been created for a list, a clients of the list will use the default policy.

When defining a list password generation policy, policies can be applied to machine types. This is called a Machine Type Filter.

The machine filter field in the password policy is a dos-style wild card string that indicates the policy should be used for any machines with a Machine Type matching the value.

Example for Windows Server 2008 R2:

- Machine type string (from Python): “Windows-7-6.1.7601-SP1”

A password policy with machine type filer of either ‘Windows-7*” or Windows*6.1*” will match the system cause the policy to be used for that system.

The machine type is a string of information that identifies the platform and system type of an endpoint machine. These strings are generated by the endpoints themselves when the endpoint code (Windows service or python script) runs. Because the Python script generates machine type information in different format than the Windows service, here is a guide of the formats generated by both endpoint types for different versions of systems.

Windows Service

Windows service endpoints only apply to versions of Windows (the Windows service should not be deployed to a non-Windows platform). Example formats replace the specific patch level with xxx because it varies.

Windows Version	Machine Type String
Vista	Microsoft Windows NT 6.0.6000
Server 2008	Microsoft Windows NT 6.0.xxx
Server 2008 R2	Microsoft Windows NT 6.1.xxx
7	Microsoft Windows NT 6.1.7600
Server 2012	Microsoft Windows NT 6.2.xxx
8	Microsoft Windows NT 6.2.xxx
Server 2012 R2	Microsoft Windows NT 6.3.xxx
8.1	Microsoft Windows NT 6.3.9600
Server 2016	Microsoft Windows NT 10.0.xxx
10	Microsoft Windows NT 10.0.10240

Python Script

The Python script applies not only to Windows, but other platforms that support Python. Not all versions and distributions are shown, but this table should provide a guide to relative formats. If there's a question about a specific format, you can investigate by either running the Python endpoint and observing what it reports, or call the Python code directly on the endpoint by starting a Python session and running

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

Example formats replace the specific patch level with xxx because it varies.

OS Version	Machine Type String
Vista	Vista-6.0.xxx
Server 2008	Vista-6.0.xxx
Server 2008 R2	Windows-7-6.1.xxx
7	Windows-7-6.1.xxx
Server 2012	Windows-post2008Server-6.2.xxx
8	Windows-post2008Server-6.2.xxx
Server 2012 R2	Windows-post2008Server-6.2.xxx
8.1	Windows-post2008Server-6.2.xxx
Server 2016	Windows-post2008Server-6.2.xxx
10	Windows-post2008Server-6.2.xxx
SLES 12	Linux-4.4.21-69-default-x86_64-with-SuSE-12-x86_64
Raspberrry Pi (Debian)	Linux-4.4.21-v7+-armv7l-with-debian-8.0
OSX (Sierra)	Darwin-13.4.0-x86_64-i386-64bit

Password Generation Settings

Simple Hashing Password Generation

The default derived password generation settings algorithm uses an MD5 hash combined with simple transforms to derive passwords with a simple fixed format. For each iteration of the password generation, the secret will be hashed using MD5 to ensure good data distribution for the resulting hash, and the resulting binary digest information will be interpreted as hex-encoded characters (32 characters). The first 2 hex digits will be replaced with the question mark symbol (?) to ensure there is a symbol in the resulting password (to satisfy password policy requirements that require a symbol). The next 14 hex characters will be lowercase numbers/letters and the remaining 16 hex characters will be uppercase numbers/letters.

For the next iteration (and subsequent iterations), the previous iteration's derived password will be used (similar to the secret in the initial calculation) to generate a new MD5 hash and the same password derivation algorithm will be applied to that hash.

Example for Simple Hashing Password Generation over 2 iterations:

Machine secret	pFKiZUQFll3yJi4H1UvEb3gTRbADrV7E
	MD5 Hash
MD5 hash for 1st iteration	359df6d30033ed522c057ce5b5bdda96
	Transform
Password transform for 1st iteration	?9df6d30033ed522C057CE5B5BDDA96
	MD5 Hash
MD5 hash 2nd iteration	964d3ffe07680f9b1696543c1dc634d8
	Transform
Password transform for 2nd iteration	?4d3ffe07680f9b1696543C1DC634D8

Defined Policy Password Generation

The defined password policy generation settings refers to passwords that are generated using specific password format settings (settings that don't generate the simple password hash). This method allows you to specify the length of the derived password, and the allowable character set (numbers, symbols, lowercase) for the resulting derived password.

For each iteration of the password generation, the secret will be hashed using MD5 to ensure good data distribution and then the resulting hash will be interpreted as a set of binary data (64 bytes of data). The set of possible ASCII characters the resulting password can contain is calculated. 95 characters are possible in total:

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

For each type of character that is not allowed, the size of the set of characters decreases. If explicit symbols sets are used, then the set size is decreased by the number of symbols that are not included in the set of allowed symbols.

Each two bytes are interpreted as a decimal number between 1 and 255 and is divided (mod) by the size of the allowable character set to get an index in the array of possible characters. The set of allowed characters is constructed by adding all upper case letters, then all lower case letters (if allowed), then numbers (if allowed), then symbols (or the allowed symbols in the order they appear in the allowable symbols string).

One character of the generated password is derived from each two bytes of the MD5 hash, up to the password length of the derived password. This means that the hash can generate enough data to derive passwords up to length 16. If a longer password is specified in the policy, then another MD5 hash is generated from the 16 character derived password, and that new hash is used to generate more characters using the same method described.

Example Policy: length 20, lowercase, numbers (no symbols)

Example available character set:

Value	Character Result
0	A
1	B
…	…
26	a
27	b
…	…
52	0
53	1
…	…
61	8
62	9

Example of a conversion from 2 byte hash value to password character:

Machine secret	BQgApSMlqsHBD3xUlK1vLbtHb2uo6Gr3
	MD5 Hash
MD5 hash	a60b978fc6d0364de97aa218d0b15272
First 2 bytes	A6(166 in decimal)
Character set size	95-33(remove symbols)=62
Character index mod	166 % 62 = 42
Character at index 42 in character set	q

Example for defined policy password generation over 2 iterations:

Machine secret	BQgApSMlqsHBD3xUlK1vLbtHb2uo6Gr3
	MD5 Hash
MD5 hash for 1st iteration	a60b978fc6d0364de97aa218d0b15272
	Convert to password characters
First 16 characters of password	qLbTMW2Pv8mYW1U0
	MD5 Hash first 16 characters
MD5 hash	70f18a8ff78f83ab64124276af3aad98
	Convert to password characters
Remaining 4 characters of password	y3OT
	Append results for password

First password (16 + 4 characters)	qLbTMW2Pv8mYW1U0y3OT
	MD5 Hash
MD5 hash for 2nd iteration	2977bfc0f1fb2fec8f8b7e501b97388d
	Convert to password characters
First 16 characters of password	p5FG3DvyTPCSbb4R
	MD5 Hash first 16 characters
MD5 hash	ae0b4ad18c6503fd7ed243cbc57c0566
	Convert to password characters
Remaining 4 characters of password	yLMX
	Append results for password
Second password (16 + 4 characters)	p5FG3DvyTPCSbb4RyLMX

Client Configuration

A newly enrolled endpoint does not need to know anything about password policy or settings, it must only know the list ID (which will default to 0) and the web service endpoint URI to download its password secret renewal policy and password policy.

Disconnected clients can be:

- Windows
- Mac OSX
- Linux

The agent is provided in two formats:

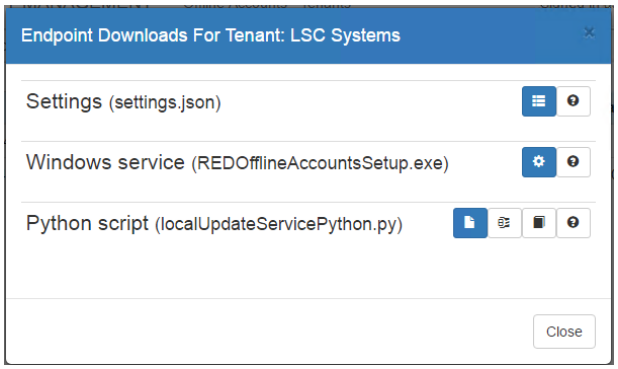
- For machines using the Windows service endpoint software, the machine must have the .Net 4.5.2 framework installed (or later).
- For machines using the Python endpoint software, anything in the Python 2.x family from version 2.6.X and up must be installed (Python 3.x is not currently supported).

Both forms of the endpoint are available as downloads from the web application or can be copied from the web application installation directory, **%inetpub%\wwwroot\pwcweb\OfflineUpdates** directory.

Client Configuration

1. Locate an existing list in the List's list and click the **download** button.
2. On the endpoint downloads page you will download two items: The **Settings** file and either the Windows Service or Python script. Download the **settings.json** file to the same location as the installer package. See "Settings.JSON" on page 16 for more information on this file.

   The Python script can be downloaded in multiple formats. If you downloaded the tar or zipped version, extract the file once downloaded.



Next install the Windows service or Python script.

Settings.JSON

The **settings.json** file that is downloaded from the web application includes the server ServiceURI and List GUID by default. If the web service is installed to a different location, the settings.json file will need to be modified to reflect that alternate URI. Following is an example settings.json file:

```
{
   "ServiceURI": "https://lsdslscprd.lsds.int:443/ERPMWebService/OfflineUpdateWebService.svc/",
   "TenantGUID": "db964ad1-05d7-4690-8800-62ec5bcb53b6"
}
```

If you want to manage an account other than the default (Administrator on Windows, root on Linux, Administrator on OSX), you can specify the name of a different local account that should be managed by the endpoint software.

To specify the name of a non-default account to be managed, add the value **LocalAccountName** to the settings.json file on the endpoint. The resulting default settings.json file will look like this:

```
{
   "LocalAccountName": "Firecall",
   "ServiceURI": "https://lsdslscprd.lsds.int:443/ERPMWebService/OfflineUpdateWebService.svc/",
   "TenantGUID": "db964ad1-05d7-4690-8800-62ec5bcb53b6"
}
```

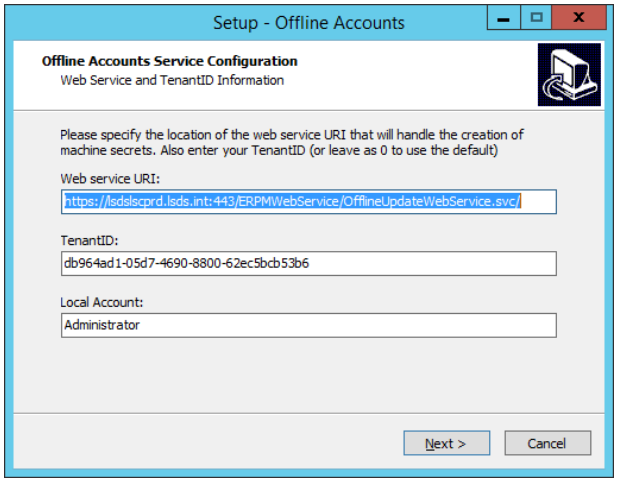
If omitted, the default account that will be managed will be Administrator on Windows, administrator on OSX , and root on Linux.

Windows Client Service Configuration

Windows Service Installation

To install the Windows service endpoint software on a windows system, download and run the Windows service installer (RedOfflineAccountsSetup.exe). Before running the Windows service installer, ensure that you have downloaded the settings.json file to the same directory and that the settings file contains the correct web service URI and List ID for the endpoint.

1. Launch the installer.
2. Verify the following information (information is supplied from the settings.json file):
   - Web service endpoint URI.
   - List ID.
   - **Local Account** - this is the name of the account to be updated.



   The web service URI and List ID can be updated later by reinstalling the client or editing the **ImagePath** value in the registry at:

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\ControlSet\Services\LiebsoftLocalUpdates
   ```

   Then restart the service.

3. Click **Next**.
4. Validate the installation location.
5. Click **Next**.
6. Choose to create a desktop shortcut if desired.
7. Click **Next**.
8. Click **Install**.
9. Choose to start the service at the end of the setup process.

The service name is **LiebsoftLocalUpdates**.

After the service is started, it will create a status.json status file in the same installation directory as the service that will contain the most recent operational status for the service endpoint. This status will also be pushed to the back end web service and available for view in the web application.

The service is set to run automatically on startup, so machines that are turned off will re-synchronize immediately once turned back on and the web service endpoint is accessible.

Note: *When first configuring the service, if the web service endpoint is unavailable, the service will start but will not randomize any passwords. It will continue to poll the web service endpoint every 60 minutes until it is successful in contacting the web service and can download its client settings. If the client has had at least one successful connection and update to the web service, the service will start in the future even when the web service is unavailable, and will randomize passwords from then on.*

Client Logging

The log file for the service is located in a logging directory under the installation directory (default is c:\program files (x86)\Lieberman\RED Offline Accounts\Logs\).

The endpoint software also attempts to connect to the web service to update the server with log messages as actions occur, but if the web service is unavailable while the endpoint is running, the log information will not be sent to the server. Logs that are sent from the endpoints to the server are accessible through the web application either at the list or machine level by clicking on the log button.

To control the logging, **LoggingThreshold** must be fined in the registry at **HKLM\SOFTWARE\Wow6432Node\Lieberman\OfflineUpdateService**:

- **0** - Trace. Warning, this level will log the password being set.
- **1** - Verbose. Logs errors, successes, and detailed messages.
- **2** - Logs success and failure messages.
- **3** - Logs error messages only.

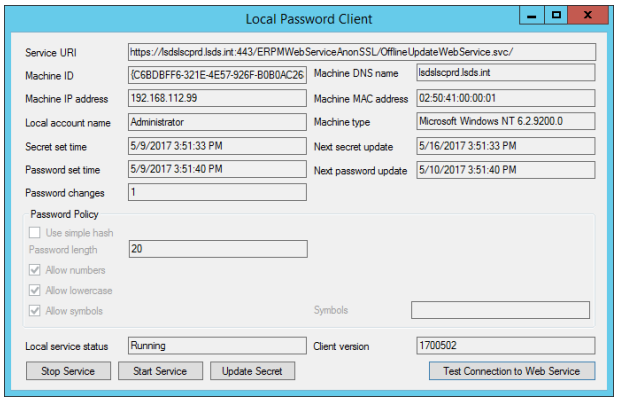
Local Password Client

To launch the console open Offline Accounts. This will require administrative permissions on the host system.

The console will not display information regarding system ID, DNS name, MAC address and more if the web service has not been connected to at least once.

The important information on this dialog surrounds the the following fields:

- **Secret set time** - The last time the client secret was set.
- **Password set time** - The last time the service reset the password.
- **Next secret update** - The next time the client secret is scheduled to update.
- **Next password update** - The next time the service will reset the password.



In addition, the client application has several command line arguments that can be provided at run time to facilitate further diagnostics. Run the following commands from an administrative command prompt:

- **LocalPasswordClient.exe WriteStatus** - Prints the most recent status of the client endpoint.
- **LocalPasswordClient.exe TestSetPassword NewPassword** - This command executes the same code the service uses when setting the password for a local account. This command can be used to diagnose problems if the service fails to update the password for the local account. Ensure that you run the client application in the same context as the service to replicate the behavior of the service. *

- **LocalPasswordClient.exe IterationTest** - Calculates and times 10,000 derived password generations. This can help determine if the password length or character set is such that it loads the system while deriving passwords.*
- **LocalPasswordClient.exe Secret** - Prints out the currently stored secret.*
- **LocalPasswordClient.exe GeneratePassword Secret 10 14 1 1 1 0** - Uses the provided settings to generate a password derived from the secret provided.*

  The input arguments (in order) are:

  - **Secret** - (string) 32 character string
  - **Iteration Count** - (int) how many iterations of derived passwords to calculate
  - **Length** - (int) derived password length
  - **Numbers** - (int) allow numbers in derived password
  - **Symbols** - (int) allow symbols in derived password
  - **Lowercase** - (int) allow lowercase letters in derived password
  - **SimpleHash** - (int) use simple hash password generation. Setting this parameter to 1 means the password length, numbers, symbols, and lowercase settings are ignored (still must be provided).
- **LocalPasswordClient.exe PasswordTest** - Prints the current derived password based on the stored settings.*
- **LocalPasswordClient.exe Password** - Prints the current derived password and secret based on the stored settings.*

* This command requires running under the same context the Windows service runs as (LocalSystem) to access the stored secret which has been encrypted with the LocalSystem's certificate. See "Local Secret Storage/Secret Protection" on page 24 for more information.

See "Retrieving Disconnected Passwords with the Windows Service" on page 30 for more information on retrieving passwords.

Windows Client Endpoint Settings

The Windows service endpoint can accept settings from several vectors. Settings can be specified through one or more of these input methods, and not all settings must be supplied using the same method. The order that settings will be used when present is as follows:

- Settings passed on the command line to the service at starts. These settings must be specified as part of the service configuration and are configured automatically as part of the service installation by the service installer. The available set of settings as command line arguments to the service are (in order they appear on the command line):
    - **Tenant ID** - The list ID
    - **Web Service URI**
    - **Local Account Name**
- Settings passed as part of the app.config file. This file is located in the same directory as the service exe and named **OfflineUpdateService.exe.config**. The available set of settings provided by the app config file are:
    - **TenantGUID** - string value
    - **ServiceURI** - string value
    - **LocalAccountName** - string value
- Settings saved to the local registry. These settings are located in the **HKEY_LOCAL_MACHINE\Software\Wow6432Node\Lieberman\OfflineUpdateService** key. The settings stored in this key preserve state of the disconnected service itself, but you can overwrite the settings (or provide additional settings) to change the configuration of the service either on startup or as it is running. Equivalent settings to those exposed on the command line and config file are values under this key:
    - **TenantGUID** - string value (not present by default, create to specify)
    - **ServiceURI** - string value
    - **LocalAccountName** - string value (not present by default, create to specify)

Python Client Script Configuration

Python Script Installation

To install the Python endpoint script, download and save (and unzip/extract) the python script to the endpoint machine. Before running the Python script, ensure that you have downloaded the settings.json file to the same directory and that the settings file contains the correct web service URI and list ID for the endpoint.

For a Linux distribution, it is recommended that you save the Python script and settings file to a path only accessible to root (like **/root/bin/REDOA** or **/bin/REDOA**) and then setup a cron job or crontab job to schedule the script to run as frequently as desired.

You can use `"crontab -e"` to edit the crontab file on endpoint machine as root (or `"sudo crontab -e"`) . Append an extra line to the file to create the new job. If you want the script to run every minute, the new line looks like this:

```
*/1 * * * * python /root/bin/REDOA/localUpdateServicePython.py > /root/bin/REDOA/cron.log
```

Note that in this example above, the path to python is set in the environment, but 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, then 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 like this:

```
*/1 * * * * python /root/bin/REDOA/localUpdateServicePython.py 0
https://lsdslscprd.lsds.int/erpmwebservice/OfflineUpdateWebService.svc/ >
/root/bin/REDOA/cron.log
```

Note that if you specify command line arguments as input to the script, those settings will be used instead of the settings specified in the settings.json file, even if the file is present.

The first part of the crontab entry line is the scheduling frequency, also specified is the path to python (version 2.6.x or higher), the path to the python update script, the ListID input argument to the update script, the web service URI to the web service, and the path to the cron log file for logging job results (optional). **Note that the trailing slash on the web service URI is required.**

Client Logging

The log file for the service is located in the installation directory and is named **localupdateservicepython.log**.

For the Python script, there are flags included in the script that control the logging threshold (verbosity and log to console output) as well as the ability to clear the log file before each run. See below for more information.

The endpoint software also attempts to connect to the web service to update the server with log messages as actions occur, but if the web service is unavailable while the endpoint is running, the log information will not be sent to the server. Logs that are sent from the endpoints to the server are accessible through the web application either at the list or machine level by clicking on the log button.

Python Script Options

The Python script when executed from the command line includes several command line arguments intended to help with health and status diagnostics of the endpoint. The command line arguments are single strings and should be the only argument passed when used, ie:

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

Available commands:

- **Version** - Print out the version of the endpoint
- **SecretAndPassword** - Print out the stored secret, the number of password generation iterations from the secret creation time and now, and the current derived password.
- **LastPassword** - Print out the last password set based on the recorded last password set time.
- **Settings** - Print out the current settings for the endpoint
- **Reset** - Clear all locally saved settings except the web service URI and list ID. This will cause the endpoint to get a new machine ID, secret, and trigger an immediate password change the next time the script is run normally.

Python Client Script Endpoint Settings

The Python script also accepts settings input through command line arguments. The order of those input arguments are:

1. **Tenant ID** - this is the list ID
2. **Web Service URI**
3. **Service Certificate file path**
4. **Local Account Name**

If no settings are specified on the command line as input, the settings.json file is used for configuration.

Endpoint Enrollment

Endpoints can be enrolled as part of an on-line process or can be pre-enrolled. This topic describes both of those processes.

Default Enrollment Process

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

The next step of machine enrollment is to gather the DNS name, IP address, MAC address, and platform information for the endpoint machine. The 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 (locally stored settings don't match what the system is reporting), the information is updated on the server through the web service.

The next step after establishing the machine ID and endpoint machine settings is to download the password and secret renewal policy for the endpoint machine. Using the machine platform type as a matching string, individual password policies can be configured for systems based on the list ID and type of system. If the system matches no specific password policy for the list, or if no password policies have been created for the list, the default password policy that is defined in the web application settings is used.

The default password policy is:

- Generate a new secret every 168 hours (7 days)
- Generate a new derived password from that secret every 24 hours
- Use the simple hashing password generation function

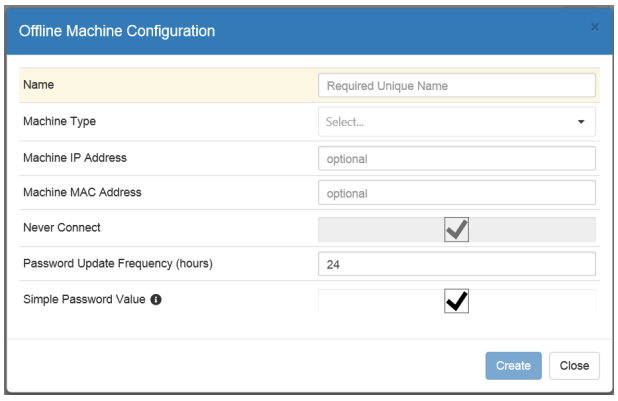
After downloading the secret renewal policy and password policy the local endpoint settings are saved.

Endpoint Pre-Enrollment

For a machine to become pre-enrolled, an entry for the endpoint must be added 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.

To pre-enroll a system...

1. In the web application, open the target list.
2. Click the **New Offline Machine** button (+).
3. Supply the following information:
    - **Name** - the desired name of the endpoint.
    - **Machine Type** - pre-configure the system type. Start typing to see other operating systems types such as Linux, OSX, etc.
    - **Machine IP Address - optional** - the IP of the system.
    - **Machine MAC Address - optional** - the MAC address of the system.
    - **Never Connect - optional** - select this option if it is known this client will never connect to the web service to update it settings and status. Selecting this option prevents mis-reporting of the client status in the web



application interface, but poses no technical problems whether enabled or not.

- **Password Update Frequency** - the amount of hours between password rotation.
- **Simple Password Value** - if selected, uses the simple password value algorithm for password generation. If unselected, you will be able to define the length and allowed characters for the password.

4. Click **Create** to add the endpoint.
5. Now download these settings and the client software to install on the client.
6. The client software will begin the randomization process and have no need to connect to the back-end web service. Ever.

Endpoint Settings

The Windows service and Python script store their local settings using different mechanisms. The Windows service creates and stores local settings in the registry under the **HKLM\Software\Wow6432Node\Lieberman\OfflineUpdateService** node. The Python script creates (or updates if an initial settings.json file was downloaded) and stores local settings in the settings.json file created in the same location as the endpoint Python script. The password policy information, as well as state information such as last secret update time and last password set time are visible in the settings. The OfflineSecret value is the only value stored in the local settings that is considered sensitive.

By default, the Local account name settings is empty, which causes the endpoint to update the Administrator account on Windows, root on Linux, and Administrator on OSX. A blank or missing list ID in the configuration will cause the endpoint to use the value 0 as the list ID. The web service URI value is a required setting, and communication to the service is verified as part of an initial endpoint enrollment, so the service URI must be correct and communication between the endpoint and the web service must be possible during initial provisioning.

Note that after initial provisioning communication between the endpoint and the web service is no longer required at any point. The endpoint will attempt to update the web service of it's status over time, but if unable to do so will continue to perform synchronous updates to the local account indefinitely based on the last synchronized settings.

Endpoint Status Settings

When the endpoint runs, the status settings for the endpoint are saved to run directory as the status.json file. This status file contains the last run time, the machine ID, the version of the endpoint software, the status code for the result of the last operation, and a status message. This information is also uploaded to the server (if the endpoint is connected to the network and the web service is reachable) and visible as the status through the endpoint machine list display.

Secret Generation

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 represented as a string (upper and lowercase letters and numbers) with a length of 32. This secret is saved on the server side and associated with the machine ID, then downloaded to and saved on the endpoint machine in the endpoint's local settings.

If a new secret cannot be generated when an endpoint secret expires based on policy, the existing secret will continue to be used to generate derived passwords until the service becomes available again and a new secret can be obtained by the endpoint. This case does not constitute a failure, although the web application will warn of an expired secret for an endpoint machine in the machine list for a list if the endpoint has not be able to connect to the service and download a new secret after the secret on the endpoint machine has expired.

Local Secret Storage/Secret Protection

The secret is the only cryptographically relevant information in this system, so protecting it is important. For the Windows service endpoint, this means running the service as LocalSystem and using ProtectedData to write the encrypted value of the secret to the

registry. For the Python endpoint, it means making the settings.json file only accessible to root for read, write, and execute (rwx) which the script does as part of it's creation or modification of the settings file (each time the Python script runs it adjusts the file permissions back to read, write, execute only for root).

Accessing Disconnected Passwords

Disconnected passwords can be accessed from the web application, web service, PowerShell or directly on the client. This section describes the various methods used and any required configurations to retrieve the password.

Granting Access to Retrieve Passwords

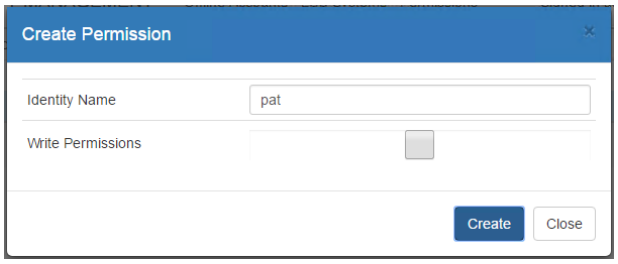
To retrieve a password from the web application or programmatically via web service or PowerShell, list delegations must be created. Permissions can be granted via the web application, programmatically via web service (but not PowerShell), or via the management console. **Permissions are granted on a per list basis, rather than on a per-machine basis.**

All access users can already retrieve any password. Users who are not all access users but require access to disconnected passwords must be granted permissions. Delegated users can only see the lists they have been delegated access for.

Lists must already exist before permissions may be granted.

Granting Permissions via the Web Application

1. Open the web application as an All access user.
2. Go to **Passwords | Disconnected Accounts**.
3. Click on the **List Permissions** button next to the desired list.
4. Click the **Add a Row** button (+) to add a new identity.
5. In the Delegation Identity column, supply the name of the identity. There is currently no browse functionality and the user name must be typed in. Any existing identity that has been granted Logon permissions can be added to the List Permissions. For example, if an identity is added as DOMAIN\LSCAdmin, that exact name must be typed in the Delegation Identity column field. If using explicit account, just type the name. See the admin guide for more information on adding identities.
6. Enable the Write Permissions permission if it is desired to allow the identity to:
    - Creating/changing password policies for the list
    - Viewing endpoint machine and password information for the list
    - Deleting endpoint machines for the list
    - Viewing log information for the list and all associated machines
    - Changing delegation on the list



7. Click **Create**.

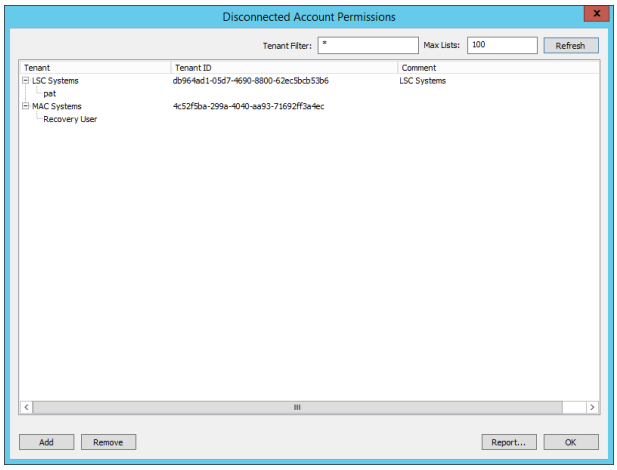
Granting Permissions Programmatically

See the programmers guide for more information.

- There is no equivalent command for PowerShell at this time.
- From SOAP, call **DelegationOps_SetPermissionOnTenant**.
- From REST, call **/REST/Delegation/Tenant**.

Granting Permissions via the Management Console

1. Open the management console.
2. Go to **Delegation | Web Application Disconnected Account Permissions**.
3. Select a list to add a delegation to and click **Add**.
4. Select one or more identities to grant permissions to and click **OK**.



Retrieving Disconnected Passwords Online

Disconnected passwords can be retrieved from the web application, programmatically or directly on the client. This section shows how to retrieve passwords from the web application or programmatically.

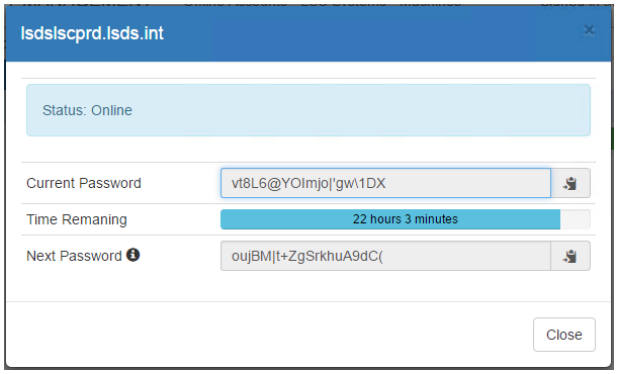
To retrieve a disconnected password requires an identity with All Access or an identity that has been delegated permissions to the list.

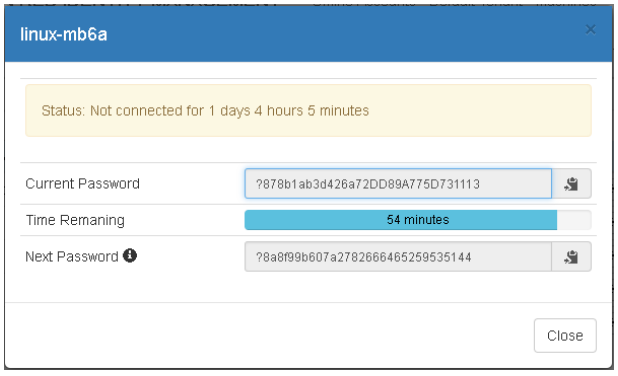
Retrieving a Disconnected Password from the Web Application

1. Log into the web application.
2. Go to **Passwords | Disconnected Accounts**. A user who is not all access will only see the list(s) for which they were previously delegated access.
3. Click the list name to open the list's list of enrolled systems.
4. Click the **Show Password** button (blue arrow) to view the current and next password.

The web application will show the current derived (expected) value of the password based on the stored secret and time between when the secret was generated and then number of iterations that have passed between that time and the current time (based on the password age policy for that machine). The password display will also show the next expected password in case there is not much time remaining on the current password.

The password display screen will also indicate the status of the machine. If the machine synchronized with the web service at the most recent expected time, then the status will be shown as Online. If the machine did not synchronize at the last expected time (which is expected for machines that are not connected to the network the web service is hosted on), the dialog will show how long the machine has been disconnected for in the status area.





Retrieving a Disconnected Password Programmatically

See the programmers guide for more information.

- There is no equivalent command for PowerShell at this time.
- From SOAP, call...
    - Current Password: **GetLocalPassword**
    - Next Password: There is no equivalent SOAP call at this time.

- From REST, call...
    - Current Password: **/REST/OfflineUpdate/Password**
    - Next Password: **/REST/OfflineUpdate/NextPassword**

Retrieving Disconnected Passwords from the Clients

The current password can be retrieved from the local clients.

Retrieving Disconnected Passwords with the Windows Service

For recovery on a Windows endpoint, run:

```
LocalPasswordClient.exe Password
```

Running this requires executing in the same security context as the service that is updating the password and storing the secret, which is LocalSystem by default. You can execute the process as local system on the endpoint if you have administrative permissions by impersonating LocalSystem or using an application like PSExec to do so. For example:

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

You can download PSExec (and related tools) from here: http://download.sysinternals.com/files/PSTools.zip

The output will be similar to this:

```
Current password:
jcr!Cl*{#;)^#l0~hr Z
Current secret:
kRqsrwwovoJqCgzt87Ji7mSNAqhKKAFy
```

Retrieving Disconnected Passwords with Python

For retrieval on a Python endpoint, run:

```
/root/bin/localUpdateServicePython.py SecretAndPassword
```

Note that this command must be run with sufficient privileges to access the settings.json file (root by default).

Removing Lists & Clients

As machines are removed from the network or as lists are no longer serviced, it will be necessary to remove their data from Privileged Identity.

This section describes the removal and cleanup process.

Removing Clients from Lists

Any "All Access" user or delegation with "write" permissions can remove a client from a list.

Clients can be removed from the web application or programmatically.

Removing a client will delete all the server-side information associated with that endpoint machine:

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

If the endpoint software is still running, the endpoint machine will be re-created the next time the endpoint synchronizes with the server, so if you intend to remove an endpoint permanently, first remove the endpoint software from the machine.

Removing a Client from the Web Application

1. Log into the web application.
2. Go to **Passwords | Disconnected Accounts**.
3. Click the list's name.
4. Locate the machine row in a list and clicking the **Delete** button (x).
5. You will be asked to confirm the deletion of an endpoint machine entry before completing the operation in the web application.

Removing a Client Programmatically

See the programmers guide for more information.

- There is no equivalent command for PowerShell at this time.
- From SOAP, call **DeleteMachine**.
- From REST, call **/REST/OfflineUpdate/Machine**.

Removing Lists

Any "All Access" user can remove a list.

Clients can be removed from the web application or programmatically.

Deleting a list will delete all the server-side information associated with the list:

- List information (tenant ID/List ID)
- Machines associated with that list

- All machine information related to machines associated with the list
- Password policy information associated with the list
- Permissions associated with the list
- All logs (from the database) associated with the list

Text log files created by the server are not deleted as part of this operation. If endpoints are still running that are using a deleted list (and the server is configured to allow endpoints to automatically create new lists), the list will be re-created with default settings when an endpoint configured to use the list next synchronizes. Existing permissions and logs as well as all other endpoint machine registrations will be deleted however.

Removing a List from the Web Application

1. Log into the web application.
2. Go to **Passwords | Disconnected Accounts**.
3. Click the **Delete** button (x) at the end of the target tenant's row.
4. You will be asked to confirm the deletion of an endpoint machine entry before completing the operation in the web application.

Removing a List Programmatically

See the programmers guide for more information.

- There is no equivalent command for PowerShell at this time.
- From SOAP, call **DeleteTenant**.
- From REST, call **/REST/OfflineUpdate/Tenant**.

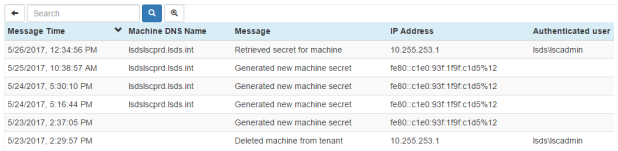
Auditing

Two sets of logs are captured for the Disconnected Account Management feature. These are not part of the standard Lieberman RED Identity Management audit logs. An All Access user or a user granted "write" permissions for the list will be able to view/export the logs.

Logs can be viewed from the web application or programmatically.

Viewing Tenant Logs from the Web Application

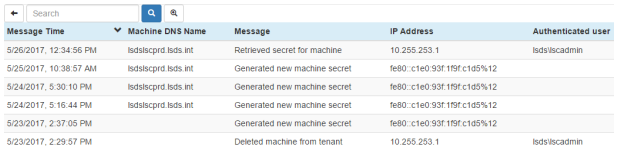
1. Log into the web application.
2. Go to **Passwords | Disconnected Accounts**.
3. Locate the target list and click **List Logs**.



The logs can be exported to CSV by clicking the **Export Data** button at the top of the page.

Viewing System Logs from the Web Application

1. Log into the web application.
2. Go to **Passwords | Disconnected Accounts**.
3. Locate the target list and click the list's name.
4. Locate the specific target machine and click **Show Logs**.



The logs can be exported to CSV by clicking the **Export Data** button at the top of the page.

Viewing Logs Programmatically

See the programmers guide for more information.

- There is no equivalent command for PowerShell at this time.
- There is no equivalent SOAP call at this time.
- From REST, call **/REST/OfflineUpdate/LogMessages**.

Notes

Duplicate System Resolution

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

OSX Endpoints

Apple OSX has a root account that does not have a password by default. The default single user installation case makes the user that is created at setup time able to access all root permissions using sudo. Users on OSX typically are managed using the dscl command to target the users in the /Users directory. Because there is no standard default user on OSX, the LocalAccountName parameter should be set in the settings.json file for OSX endpoint machines to target a specific user on that endpoint machine. The Python script can still be run as root on OSX (and thus prevent other users from being able to read/write/execute the files without root permissions) by installing the script using sudo.

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. As a result, the validity of the server's SSL certificate will not be verified (if SSL is enabled on the web service endpoint) when Python 2.6 is used by a client endpoint. The endpoint will also fail if the web service server requires SSL functionality such as client certificate validation or Windows Integrated Authentication.

Additional Resources

- **.NET ProtectedData**

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

- **Sysinternals PSExec**

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

Privileged Identity Limited Warranty

The media (optional) and manual that make up this software are warranted by Bomgar Lieberman to be free of defects in materials and workmanship for a period of 30-days from the date of your purchase. If you notify us within the warranty period of such defects in material and workmanship, we will replace the defective manual or media (if either were supplied).

The sole remedy for breach of this warranty is limited to replacement of defective materials and/or refund of purchase price and does not include any other kinds of damages.

Apart from the foregoing limited warranty, the software programs are provided "AS-IS," without warranty of any kind, either expressed or implied. The entire risk as to the performance of the programs is with the purchaser. Bomgar Lieberman does not warrant that the operation will be uninterrupted or error-free. Bomgar Lieberman assumes no responsibility or liability of any kind for errors in the programs or documentation of/for consequences of any such errors.

This agreement is governed by the laws of the State of California.

Should you have any questions concerning this Agreement, or if you wish to contact Bomgar Lieberman, please write:

Bomgar Lieberman

1875 Century Park East, Suite 1200

Los Angeles, CA 90067

Privileged Identity License Agreement

This is a legal and binding contract between you, the end user, and Bomgar Lieberman. By using this software, you agree to be bound by the terms of this agreement. If you do not agree to the terms of this agreement, you should return the software and documentation as well as all accompanying items promptly for a refund.

1. Your Rights: Bomgar Lieberman hereby grants you the right to use RED Systems Management to manage the licensed number of systems purchased. This software is licensed for use by a single client and its designated employees, contractors and authorized 3rd parties to manage the systems owned/used by a single client. The software license may not be shared with unrelated 3rd parties.

   The serial number provided by Bomgar Lieberman is designed for installation on a specific machine. You many install an unlimited number of copies of RED Systems Management for your administrators that connect to the single licensed machine. All administrators can share the pool of purchased managed node licenses.

   There are no limits to the number of web servers or clients that may access the data stored by your licensed copy of RED Systems Management. You may install and use the “RED Systems Management: Web Interface to Random Password Generator Password Recovery Console” with your duly licensed copy of RED Systems Management + Random Password Generator without any additional payment to Bomgar Lieberman.

   The cost of Microsoft web servers, SSL certificates, and other supporting equipment and technology are the sole responsibility of the user of this software-not Bomgar Lieberman.

2. Copyright. The SOFTWARE is owned by Bomgar Lieberman and is protected by United States copyright law and international treaty provisions. Therefore, you must treat the software like any other copyrighted material (e.g. a book or musical recording) except that you may either (a) make one copy of the SOFTWARE solely for backup and archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup and archival purposes. The manual is a copyrighted work also--you may not make copies of the manual for any purpose other than the use of the software.
3. Other Restrictions: You may not rent, lease, or transfer the SOFTWARE to any other entity. You may not reverse engineer, de-compile, or disassemble the SOFTWARE that is provided solely as executable programs (EXE files). If the SOFTWARE is an update, any transfer must include the update and all prior versions.
4. Notice: This software contains functionality designed to periodically notify Bomgar Lieberman of demo usage and of the detection of suspected pirated license keys. By using this software, you consent to allow the software to send information to Bomgar Lieberman under these circumstances, and you agree to not hold Bomgar Lieberman responsible for the use of any or all of the information by Bomgar Lieberman or any third party.

   When used lawfully, this software periodically transmits to us the serial number and network identification information of the machine running the software. No personally identifiable information or usage details are transmitted to us in this case. The program does not contain any spyware or remote control functionality that may be activated remotely by us or any other 3rd party.

Bomgar Corporation

578 Highland Colony Parkway

Ridgeland, MS 39157

866.205.3650

Support: help.bomgar.com

Website: www.bomgar.com