HITACHI
Inspire the Next

Bravura Privilege 12.2.4

Blue Prism Integration

Software revision: 12.2.4
Document revision: 30072
Last changed: 2022-03-01

Contents

List of Figures

List of Tables

Blue Prism Integration 1

1.1 Introduction

Hitachi ID Bravura Privilege secures access to elevated privileges. It eliminates shared and static passwords to privileged accounts. It enforces strong authentication and reliable authorization prior to granting access. User access is logged, creating strong accountability.

Bravura Privilege secures access at scale, supporting over a million password changes daily and access by thousands of authorized users. It is designed for reliability, to ensure continuous access to shared accounts and security groups, even in the event of a site-wide disaster.

Bravura Privilege grants access to authorized users, applications and services. It can integrate with every client, server, hypervisor, guest OS, database and application, on-premises or in the cloud.

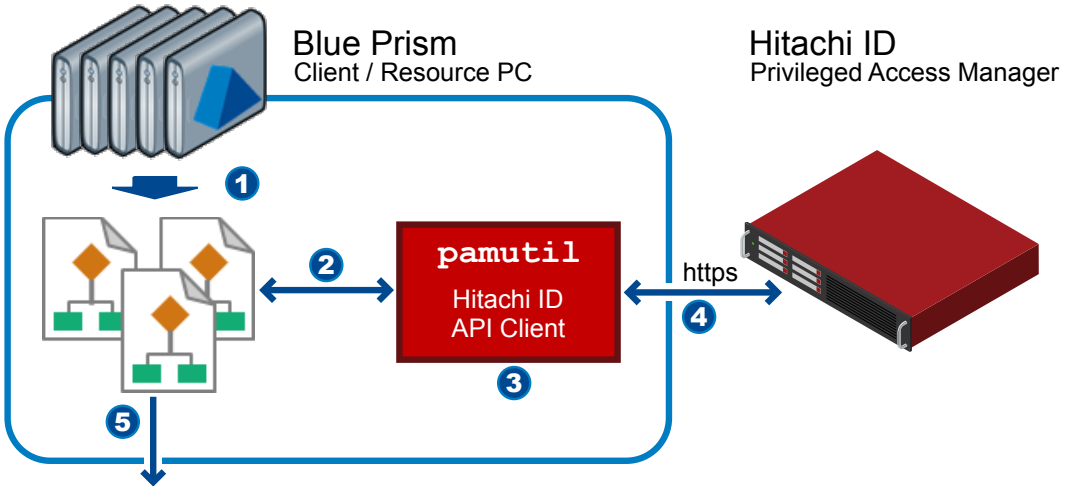
Bravura Privilege can be used to secure credentials from Blue Prism processes used to automate BPA tasks and has multiple options for securing callers of credential access requests.

1.2 Architecture

- **`pamutil`** is a collection of libraries and utilities that mediate access to *Bravura Privilege* servers. One or more **`pamutil`** clients need to be installed on *resource PCs* where credentials will be used.

  See Using runwithpass / pamutil for more information.

- The Blue Prism Utility Environment can be used to call the **`runwithpass`** executable provided as part of the **`pamutil`** package.



In the above diagram:

1. Blue Prism resources execute processes.
2. A Blue Prism process requests credentials from **`pamutil`**.
3. The “fingerprint” of the requesting caller/environment is obtained.
4. *Bravura Privilege* returns credentials to process through **`pamutil`**.
5. Blue Prism processes can automate logins into applications with credentials.

Refer to *Securing Privileged Access with Hitachi ID Privileged Access Manager* for more details.

1.3 System Requirements

- *Hitachi ID Bravura Privilege* version 9.*x* or higher.
- Blue Prism 6.*x*

1.4 Usage scenarios

The following are typical scenarios for Blue Prism integration:

1. Providing Blue Prism automation targets with credentials.

   The primary scenario shows how automation processes can obtain credentials from *Bravura Privilege*. These can be used to connect to accounts on systems where the credentials are managed by *Bravura Privilege*.

2. Running operating system processes as part of Blue Prism automation.

   Some actions (typically calling command-line programs) can have credentials passed directly into them on the command line or replaced in files immediately before the command is executed.

3. Controlling and starting Blue Prism automation processes.

   Blue Prism programs can accept credentials on the command line. These can be started with credentials stored by *Bravura Privilege*.

1.5 Integration components

The only components that are required to access credentials from *Bravura Privilege* are an installed and configured **`pamutil`** client and the Blue Prism Utility Environment. The **`pamutil`** client is ordinarily installed by the administrators of the resource PC.

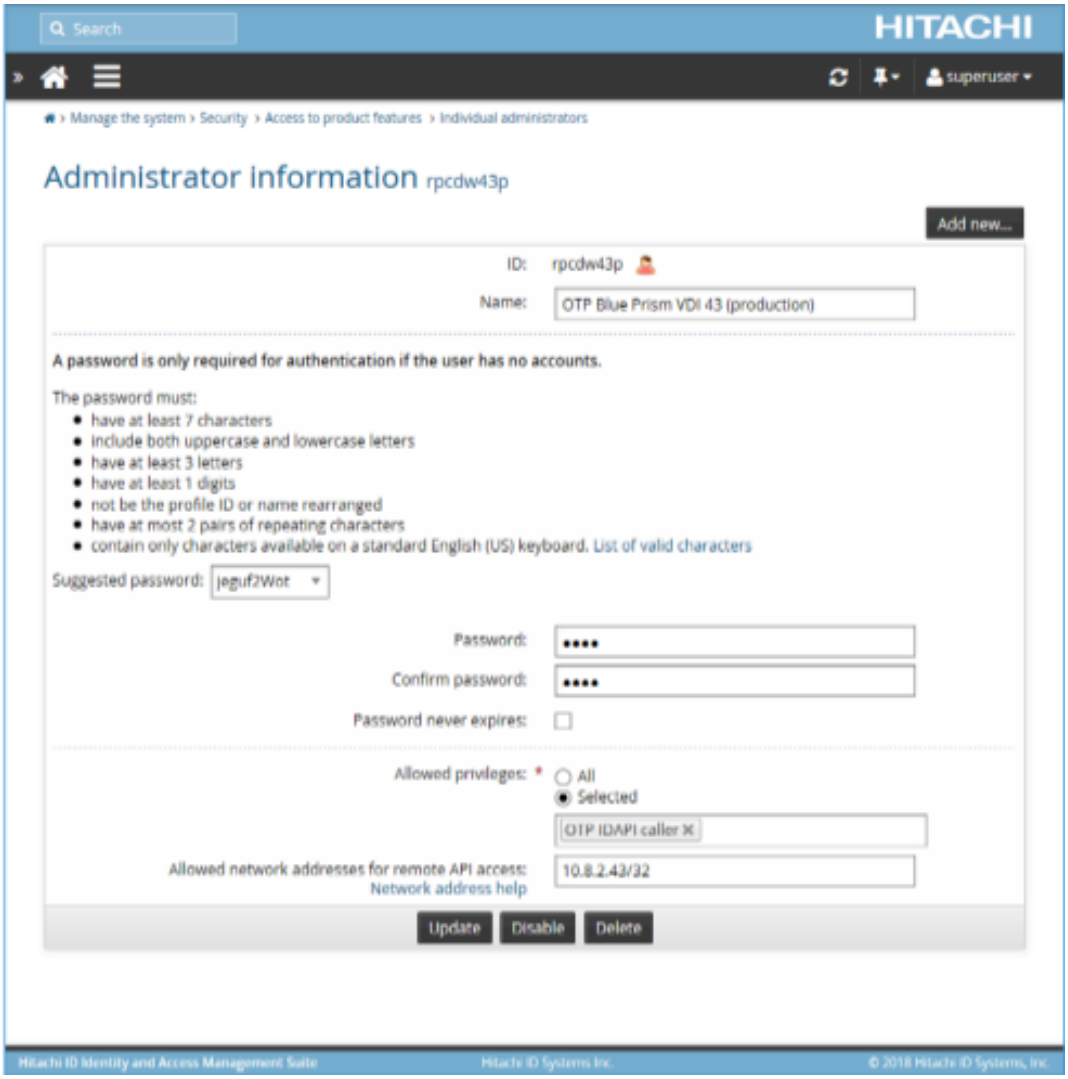
1.6 Initialization

Before **`pamutil`** can be used to retrieve credentials it must be initialized. In controlled environments this is normally a system administration task. To initialize a **`pamutil`** environment you need:

1. An initial OTP API user to access the API.

   These can be provisioned through a predefined request workflow or created on an individual basis by administrators. The OTP API user will also need permissions to access the resources and accounts it will need.

   Following is an example of an OTP API user:



Configuring OTP API Users via the *Manage the system* (PSA) module describes how to configure an OTP API user via the *Manage the system* (PSA) module.

Creating an OTP API User with Bravura Pattern Privilege describes how to create an OTP API user with *Hitachi ID Bravura Pattern: Privileged Access Edition*.

2. The **`pamutil`** configuration file (by default, named `config.ini`) which specifies the API connection parameters and details.

3. A prepared credentials file which will contain the encrypted state and account cache information. The default name for this file is `creds.ini`

   The command `runwithpass -initial` will prompt for the OTP API account's initial password, update the credentials file and initialize the encryption. At this point Blue Prism can use credentials stored within *Bravura Privilege*.

1.7 Fingerprinting Callers

The **`pamutil`** client has the ability to "fingerprint" both the caller and the calling environment. This can help reduce the risk of unintended access disclosure – if the initialization fingerprint is different than the caller's, then access to the credential is denied. Fingerprinting is optional but it can be applied as necessary in high-risk environments. It can also help to prevent unintended configuration mishaps from occurring in dynamic large scale deployments.

If you want to prevent credential access from unauthorized systems that simply had copies of the **`creds.ini`** file, you can add the line:

```
usemachinekey=1
```

to the `config.ini` before it is initialized. Callers on other machines will not be able to obtain the credentials if the machine attributes (such as the MAC address) are different than the one it was initialized with.

Similarly, the line:

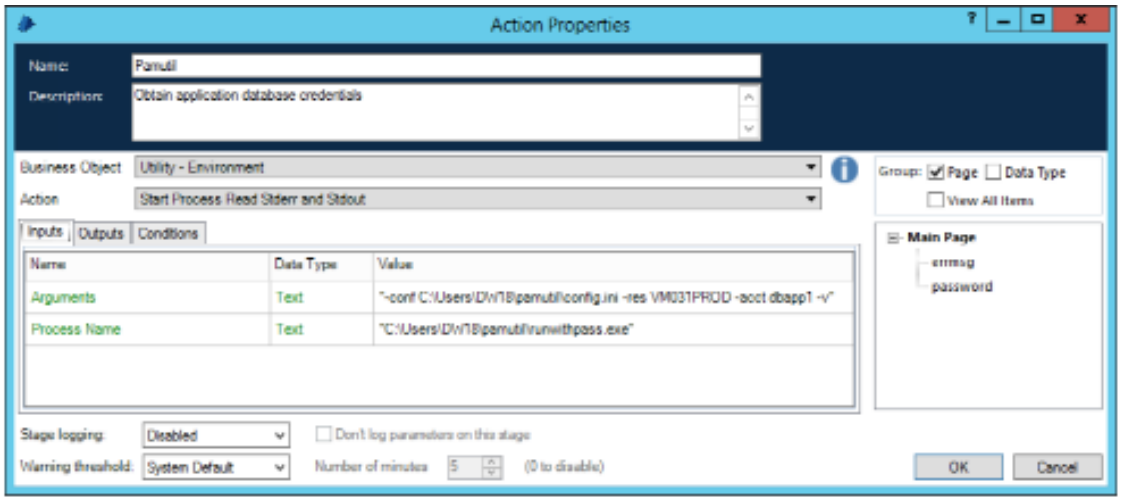
```
filekey=./config.ini
```

Will prevent disclosure of credentials when its own configuration has been modified. In both cases, **`pamutil`** needs to be re-initialized with a new OTP API password before credentials can be retrieved.

See Configuration and credentials files for additional methods of fingerprinting **`pamutil`** callers in your version of *Bravura Privilege*.

1.8 Obtaining Credentials

After initialization, Blue Prism developers can obtain the credentials by calling **`runwithpass`** from the "Utility - Environment" "Start Process Read Stderr and Stdout" Action as shown below:



This will retrieve the password for the "dbapp1" account on the "VM031PROD" resource and places it into the data item "password."

Additional calling considerations:

- Multiple names or locations for the `config.ini` configuration file can be specified. Each configuration specifies the location of its credential storage file `creds.ini`.
- You should avoid obtaining the credentials as a data item unless it is absolutely necessary. The **`runwithpass`** program can also execute commands directly without retrieving them.
- Obtain credentials only immediately before they are used (as late as possible). If they are not used again, then re-assign (or clear) the variable values after use as a precaution.
- In cases where it is necessary to obtain sensitive information from **`pamutil`** then the Blue Prism `Password` data type should be used.
- In production, you may want to ensure that logging is *turned off* for this stage. This will help contain the inadvertent release of sensitive information through diagnostic channels.

1.8.1 Use case

The following steps demonstrate how to obtain credentials using Blue Prism:

1. Set up an OTP IDAPI caller.

   You can either use the *Manage the system* (`PSA`) module or with request workflow.

2. Configure **`pamutil`** with the following command:

   ```
   runwithpass.exe -initial
   ```

3. Make sure **`runwithpass`** can retrieve a managed account password correctly.

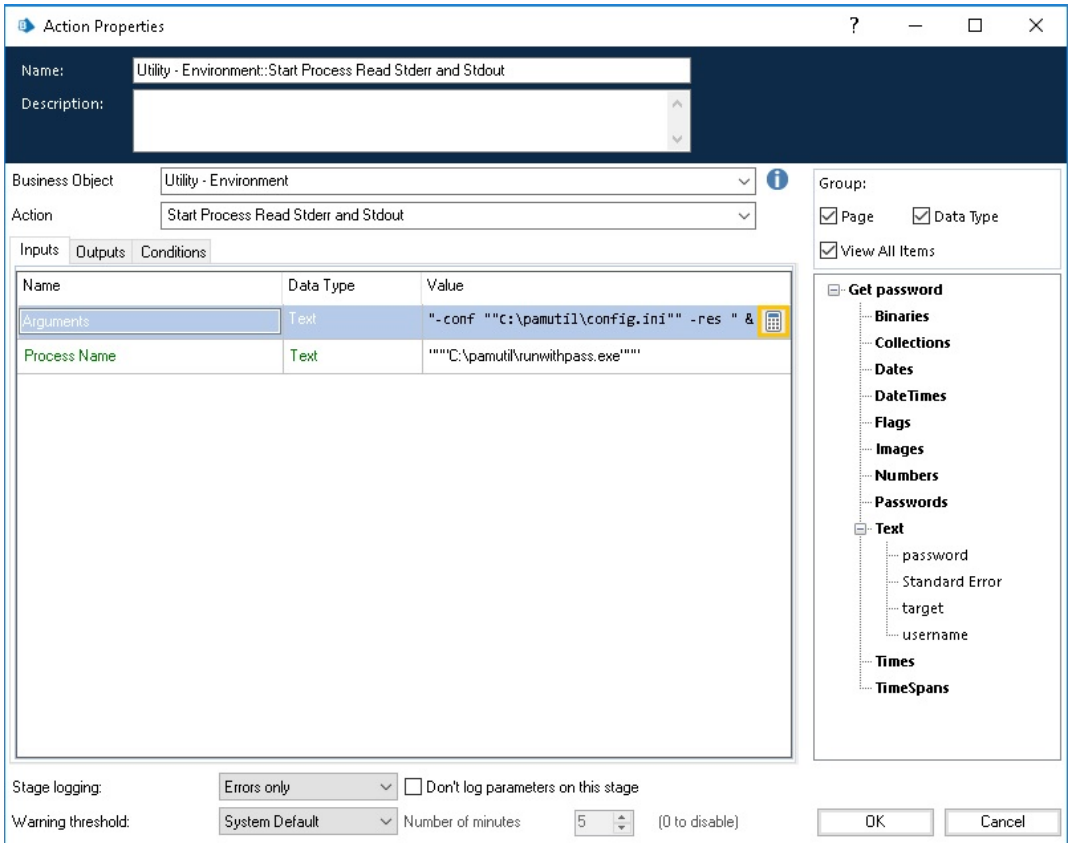
4. To use **`pamutil`** in BluePrism, grab the environment-utility from

   `https://github.com/blue-prism/environment-utility`

   or use a business object that can read standard-out / standard-error.

5. Create an **Action** to use the business object to run **`runwithpass`**.

6. Configure the business object to run **`runwithpass`** as shown in the example below:



For the **Business Object** "Utility - Environment", use the **Action** "Start Process Read Stderr and Stdout" with the inputs:

Arguments

```
"-conf ""C:\pamutil\config.ini"" -res " & [target] & " -acct " & [username]
```

Process name

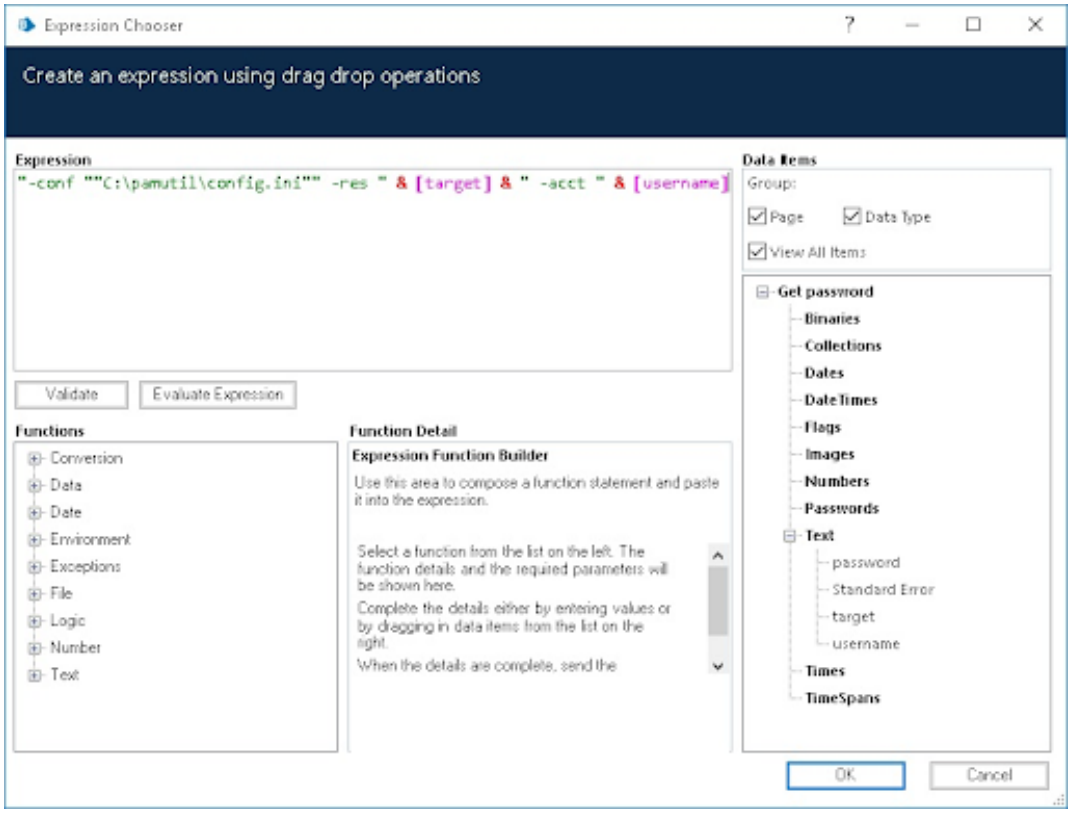
```
"""C:\pamutil\runwithpass.exe"""
```

We need to specify the path to **`runwithpass`** and the **`config.ini`** file, where:

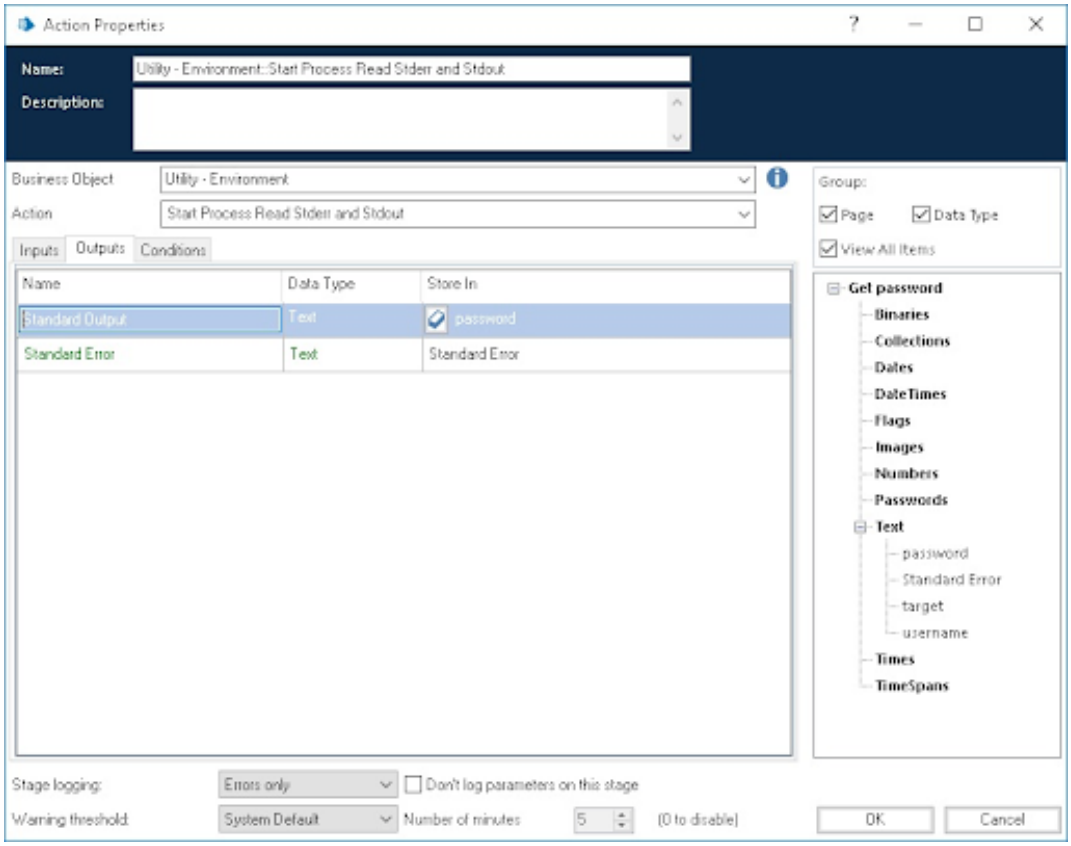
target is the managed system ID

username is the managed account

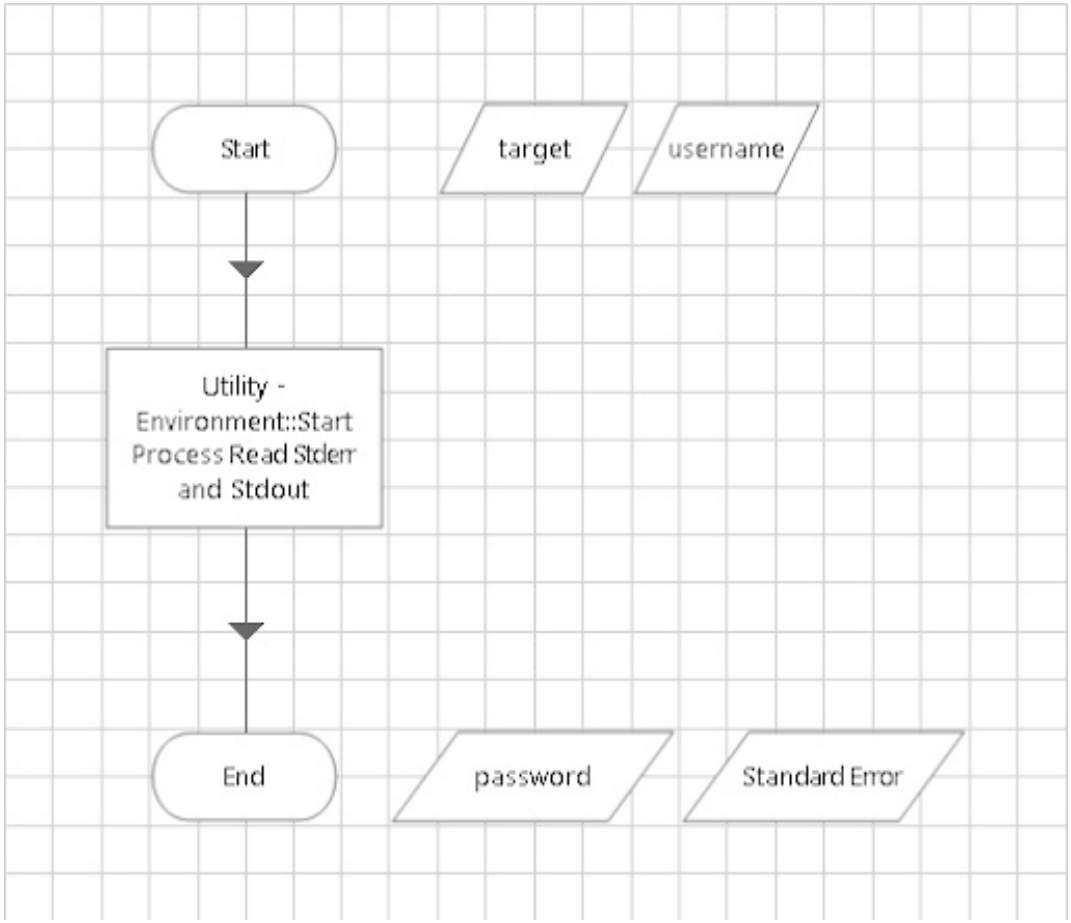
target and **username** are in this format because they are Blue Prism Data Items and the `&` is used to concatenate the data item into the Arguments input.



7. For the **Outputs**, store Standard Output and Standard Error into Data Items:

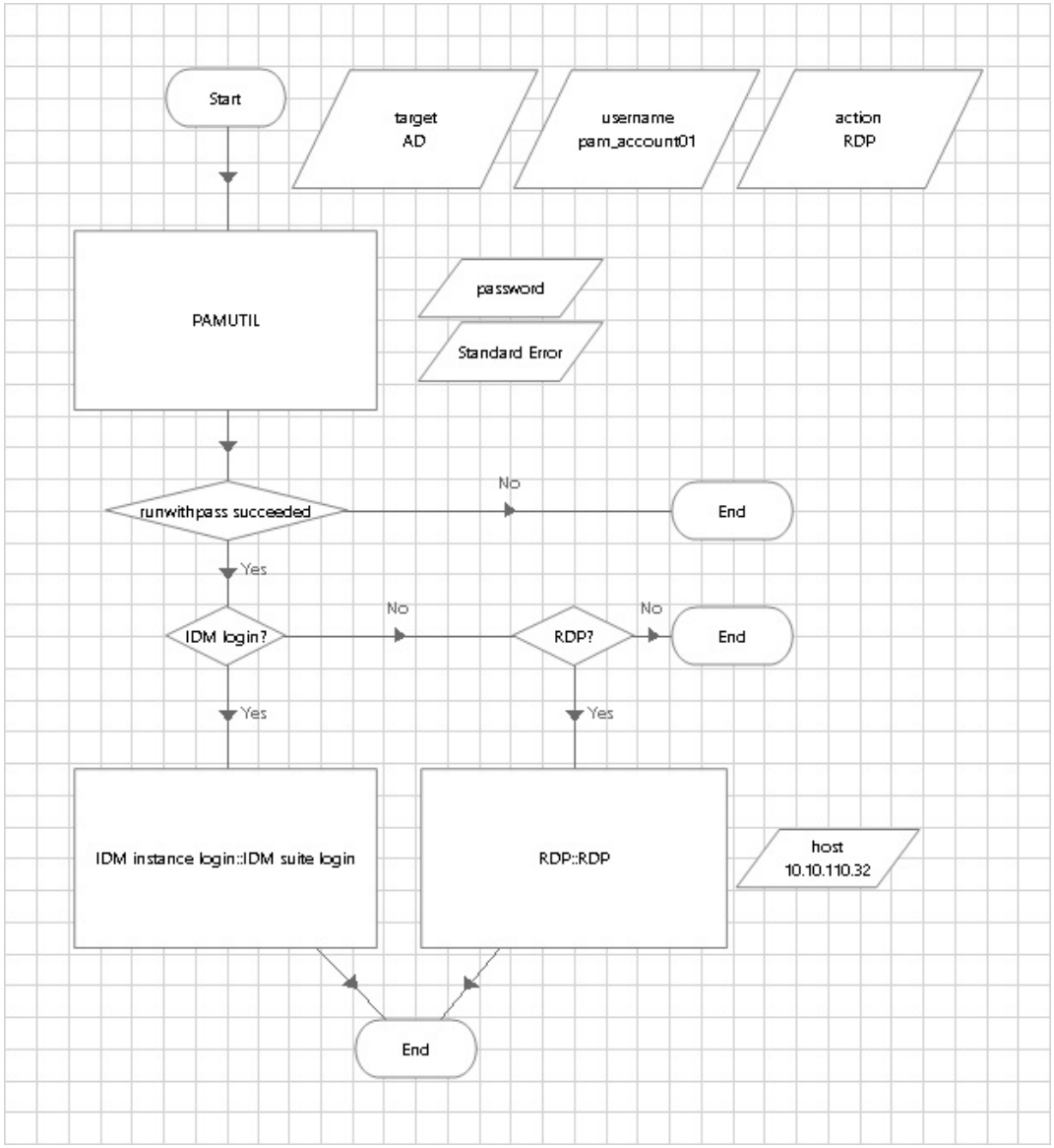


8. Once the action is configured with the business object to run **`runwithpass`**, we can set the Data items for the inputs and run the process to verify that the password can be retrieved without errors.



9. We can use the **`pamutil`** / **`runwithpass`** action with other business objects or publish this business object to use with as a business object as a whole.

   In the following example, we show how we can use **`pamutil`** to either login to a *Bravura Privilege* instance or RDP into a workstation using the managed account and its password:



1.9 Running Processes without handling credentials

It is not always necessary to handle the credentials themselves. Since **runwithpass** can send the credentials into the command directly without retrieving them into data items, it can be used for simple tasks that require an account password. Here is an example that ensures that a drive mapping exists:

```
runwithpass -res fs02 -acct bk01 -keyword X -- net use J: \\fs02\bak /user:bk01 X"
```

1.10 Starting Automation Processes

Some Blue Prism utilities have command line arguments that accept credentials. Instead of leaving the command line parameters exposed, the passwords should be stored in *Bravura Privilege* and retrieved when the utility is started.

```
runwithpass -conf config.ini -res rpc04 -keyword PWD -acct dw09 --
AutomateC.exe /run myprocess /resource rpc04 /user dw09 PWD
```

In this example, **pamutil** will retrieve the password for the **dw09** account on the machine (or resource) rpc04 and call AutomateC to start "myprocess" replacing the PWD field with the actual password.

Configuring OTP API Users via the *Manage the system* (PSA) module

2

To configure *Hitachi ID Bravura Privilege* access to privileged access API functions:

1. Create an `_OTP_USER` product administrator account with the **OTP IDAPI caller** administrative privilege.

   The **IP address with CIDR bitmask** field must specify the list of IP addresses from which the product administrator will access the API Service (idapi).

2. Create a user class with the following properties:

   - ID: _EXPLICIT_OTP_USERS_
   - Participants: USERID
   - Explicit user: _OTP_USER

3. Create a user group

   - ID: _OTP_USERGROUP
   - Access control: For the managed system from which you are requesting passwords, grant **Pre-approved check-out of managed accounts** and **Request check-out of managed accounts**
   - membership criteria: _EXPLICIT_OTP_USERS_

   user groups.

Creating an OTP API User with Bravura Pattern Privilege

3

Users with the **OTP Trustees** privilege can create OTP API users who have the permission to retrieve managed account passwords from the *Hitachi ID Bravura Privilege* vault and use them to execute scripts and command-line programs using runwithpass/pamutil.

OTP trustees have access to the following pre-defined request:

- **PAMUtil: Create OTP API User**

3.1 Use case: Create an OTP API user

This use case demonstrates how to create an OTP API user that can be used to retrieve managed account passwords using **`pamutil`**.

Assumptions

In order to create an OTP API user, at least one account is onboarded by an account trustee, or a vault account is created by a vault trustee. See the *Bravura Security Fabric* Documentation .

3.1.1 Create an OTP API group

1. Log into *Bravura Privilege* as a team trustee for the corporate AD accounts team.
2. Click **Manage resources**.
3. Click the **Team: Update** pre-defined request (PDR).
4. Select the **Corporate AD Accounts** team.
5. Add a new group called `OTP API Trustee`.
6. Provide this group with OTP trustee privileges.
7. Submit the request.
8. Use the **Team: Management Group Membership** PDR.

9. Add a user to the **OTP API Trustee** group.

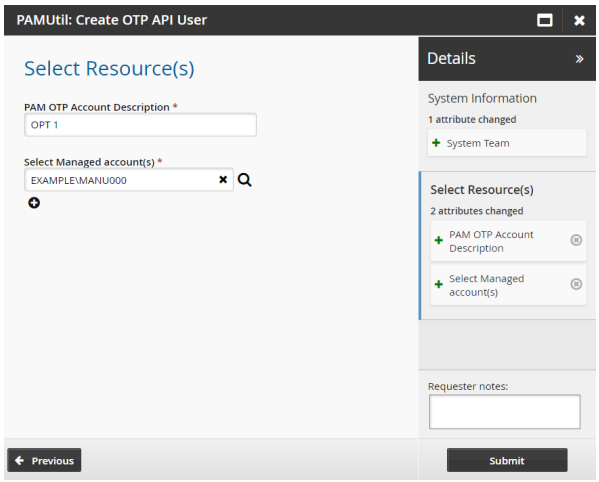
> **Note:** OTP trustees are users who can make OTP API account management requests.

3.1.2 Create an OTP_User

1. Log into *Bravura Privilege* as a user from the OTP API Trustee group created in Create an OTP API group.
2. Click **Manage resources**.

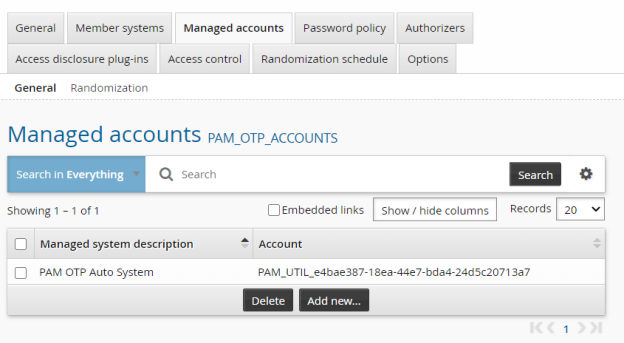
> **Note:** Only an OTP API trustee can see the **PAMUtil: Create OTP API User** PDR.

3. Click the **PAMUtil: Create OTP API User** PDR.
4. Choose the **System Team**.
5. Click **Next**.
6. Add a **PAM OTP Account Description**.
7. Select the managed accounts.



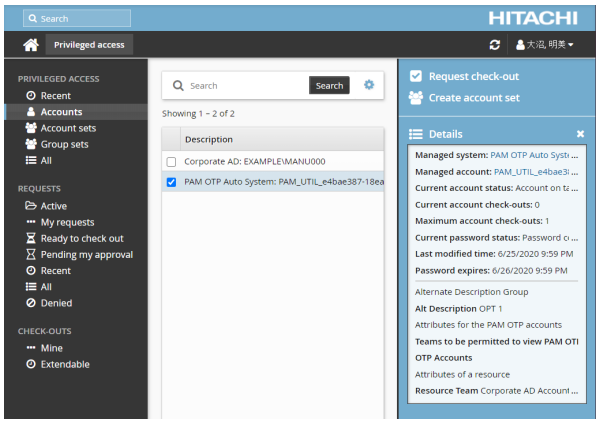
8. Click **Submit**.

The **PAMUtil: Create OTP API User** PDR creates a managed account that appears in the PAM_OTP_ACCOUNTS managed system policy in the format of PAM_UTIL_<*guid*>, and a product administrator profile to manage the account.



3.1.3 Check out the account

1. Log into *Bravura Privilege* as a user from the OTP API Trustee group created earlier.
2. Click **Privileged access** from the **Privileged access to managed systems** section.
3. Click **Accounts** from the Filter panel.
4. Select the **PAM OTP** account from the Results panel.



5. Click **Request check-out**.
6. Click **Submit**.
7. Authorize the request.
8. Click **Check out** from the Actions panel.

   The PAM OTP account can now be used by the product administrator profile created earlier, to retrieve credentials of managed accounts it has access to using pamutil.

> **WARNING!:** Do not check out the vault account again because it will be randomized on check-out and then the one used with **`pamutil`** will be wrong as it does not get updated with the new, randomized password.

Using runwithpass / pamutil

4

Description

The **runwithpass** program allows you to securely download large credential files, as well as retrieve privileged passwords from the *Hitachi ID Bravura Privilege* vault and use them to execute scripts and command-line programs.

The **pamutil** shared library is a library that provides the same functionality but is intended to be invoked from locally compiled programs; that is, it allows a program to fetch a password on demand.

To use the program or library, you must:

1. Configure product administrator access.
2. Copy the program or library, and configuration files, to the system on which you will run scripts or commands.
3. Edit the configuration file.

4.1 Installing files

Copy the appropriate program or library, and configuration files, to the system on which you will run scripts or commands.

4.1.1 Windows systems

For deployment on Windows systems, the program, library and configuration files are located in zip files in the util and utilx64 directories. Use the package appropriate for your system and needs:

- **pamutil(-x64).zip** which includes common files for deployment on Windows systems
- **pamutil-devkit(x64).zip** which includes additional files for developers writing programs that link against **pamutil**

4.1.2 Unix/Linux systems

For deployment on Unix/Linux systems, use the distribution's package manager to install the hid-common, hid-idapi and hid-pamutil packages found in the Unix/Linux *tar.gz archives in the <*instance*>\addon\idmunix directory. Once the packages have been installed the program, library and configuration files are typically located in the /usr/local/psunix/ directory.

Hitachi ID Bravura Privilege includes binaries and library files for a variety of Unix and Linux variants. Contact support@Hitachi-ID.com for additional variants.

Install the following shared object libraries first, which are required by **`pamutil`**.

- openssl – available from http://www.openssl.org/ (version 1.1.x has been verified on supported platforms)
- libcurl – available from http://curl.haxx.se/libcurl/ (version 7.26.0 has been tested on supported platforms)

It is *not* recommended to deploy **`runwithpass`** without SSL.

4.2 Configuration and credentials files

Both the program and library use the following files:

- The configuration file, supplied as **`config.ini`**, which defines configuration parameters.
- The credentials file, which is specified in the configuration file, holds encrypted passwords.
- The large credentials configuration file, supplied as **`lcinfo.ini`**, which contains information about large credential files.

These file can be renamed and can be placed anywhere on the filesystem.

4.2.1 Configuration file

The configuration file, typically called **`config.ini`**, specifies the following parameters, in simple 'key=value' format:

apiurl=https://<*server*>/<*instancename*>/idapi The URL to the *Hitachi ID Bravura Privilege* API web service.

Edit the value to suit your environment.

proxy= An optional proxy to use when accessing the API SOAP Service (idapisoap) endpoint. If omitted, uses the default proxy from environment variables on Unix/Linux system or the proxy settings in user registry on Windows. If empty, disables all proxy use

capath= For Unix/Linux systems only: The CA directory or file holding the certificates to trust.

cert= The certification file in PEM format for client authentication.

> **WARNING!:** This is *not* recommended for use without supervision from Hitachi ID Systems staff.

ignorebadservercert=0 Whether to ignore problems with server certificate and identity validation. The value in the shipped file is `0` (do not ignore).

timeout=30 The API SOAP Service (idapisoap) call timeout in seconds.

lockfile=/tmp/.pam-lock The filesystem path where a lock will be placed while calling the web service, to ensure that two processes do not simultaneously access the same one-time password (OTP) (p21), which could lead to an invalid OTP being retained locally.

locktries=200 The number of tries to lock the file.

locksleep=0.1 The delay between attempts to lock the file, in seconds.

credsfile=creds.ini The filesystem path where credentials will be stored

lcinfofile=lcinfo.ini The filesystem path where large credential file information will be stored

cacheseconds=60 The lifetime of cached credentials

usemachinekey=1 Whether to include MAC address, IP address and host name in the encryption key. The value in the shipped file is `1` (do include).

useargskey=1 Whether to include the command line details to the encryption key.

synchronouswrite=0 Whether to wait and see if operations, such as randomize and override, finish and return the result or not. The value in the shipped file is `0` (do not wait).

filekey=./config.ini

filekey=/usr/local/lib/libidapi.so The files to include into the encryption key.

4.2.2 Credentials file

The credentials file, specified in the configuration file (**`config.ini`**), holds two types of passwords:

- A one-time password, used by **`runwithpass`** and/or pamutil shared object to authenticate to the *Hitachi ID Bravura Privilege* web service.
- Cached copies of the passwords fetched from the *Bravura Privilege* web service.

The file has one line of text per system/account/password. For example:

```
system=__OTP__|user=ID|password=PW|expires=0
```

Where:

- `ID` is the ID of the product administrator (OTP), proxy user, or managed account ID.
- `PW` is the password for this ID

The file will be created upon initializing the one-time password user, as detailed in One-time passwords.

4.2.3 Large credentials configuration file

The **`lcinfo.ini`** configuration file contains information about large credential files. This file is created upon the first download of a large credential file using **`pamutil`**. Each time a new large credential file gets downloaded, its information gets recorded into this configuration file.

Installation and setup of *Hitachi ID Bravura Pattern: Privileged Access Edition* is required in order to upload vaulted files. See the *Bravura Security Fabric* Documentation for more information.

The file has one line of text per large credential file. For example:

```
system=TESTSYS|user=sampleCredFile|attrkey=LC_FILE|filename=sampleCredFile.zip|lchash=
 zkYU108lTR10b8EGgxLDvQ==
```

Where:

- `system` is the ID of the vault system or team vault the file is associated to
- `user` is the ID of the vaulted file
- `attrkey` is the attribute key the file is added to (typically LC_FILE)
- `filename` is the name of the file
- `lchash` is the hash of the file contents

This configuration file does not store password information.

4.3 One-time passwords

The *Hitachi ID Bravura Privilege* API can eliminate static, plaintext passwords embedded in scripts and configuration files. This is done by creating one user ID per application per server; that is, an application running on 10 servers requires 10 user IDs. These IDs are *Bravura Privilege* product administrator IDs with the "OTP IDAPI caller" privilege, and granted access to just the passwords they need to retrieve, randomize, or override.

OTP IDAPI account IDs are subject to two extra authentication constraints, as compared to human *Bravura Privilege* users:

- They must authenticate to the *Bravura Privilege* web services API with a one time password; whenever an OTP IDAPI ID successfully signs into the API, it uses the previous passwords but receives a new password, to be used next time.
- They must authenticate from a previously defined IP subnet; that is, they must be connecting from a well known application server.

This means that any user of the API has to be initialized with an application ID's password and must track changes to that password on every call to the API. In turn, this means that API access should be serialized, to avoid a race condition where two processes call the API using the same application ID from the same machine at the same time, and it's not clear which new password is the most current one.

The **`runwithpass`** program and the **`pamutil`** library shared object take care of recording changes to the OTP and serializing API access via a lock file.

One-time passwords and cached passwords are both stored in the **`creds.ini`** credentials file. The file name may vary, as specified in the configuration file. The ID and password must be initialized before it can be used by this program to connect to the *Bravura Privilege* web services.

To initialize the **`creds.ini`** credential file with an ID and password use the following command:

```
runwithpass -initial
```

Enter the ID and password as prompted.

Alternatively, the **`pamutil API`** function **`SetInitialPasswords`** can also be used. See pamutil API in the Bravura Security Fabric *Reference Manual* for details.

> **Note:** If the `useargskey` option is enabled in the **`config.ini`** all other arguments that would be normally used need to be specified. See Using useargskey to tie API account, managed account, and command together in the *Reference Manual* for more information

In the scenario where the OTP passwords needs to be reset the previously mentioned command can be used to reset the **`creds.ini`** credential file.

Each time an OTP IDAPI account signs into the API, its password gets randomized. You can set how often the password gets randomized, in hour intervals. The option, **Number of hours between password**

randomizations, is available for *Bravura Privilege* product administrator IDs with the "OTP IDAPI caller" privilege, located in **Manage the system** → **Security** → **Access to product features** → **Individual administrators**.

By default, passwords that OTP IDAPI accounts use to sign into the API are immediately discarded. However, you can set a limit of how many passwords previously used by an OTP account that will be considered valid when authenticating to the API. These passwords will be stored in a password history, and the oldest password will be removed each time the OTP account signs in again. This option is configured from **Manage the system** → **Security** → **Options** → **OTP MAXIMUM**.

4.4 Key management

The credentials file, **`creds.ini`**, contains both cached passwords fetched via the *Hitachi ID Bravura Privilege* API and the current value of the one-time password. This file needs to be protected, which in practice means encrypted.

The question then is how to synthesize an encryption key to use when encrypting passwords in this file? A plaintext key is clearly not desirable, since the whole point of the API is to eliminate plaintext passwords (keys and passwords are essentially interchangeable).

The only realistic option is to have the API wrapper synthesize a key from characteristics of the runtime environment. This way, if the credentials file is moved to another machine or if an attacker gains partial access to the server or its filesystem, the key generation process would yield a new key and so cached credentials and the OTP will become unavailable.

The **`runwithpass`** program and the **`pamutil`** library support a number of inputs into the key generation process:

- Characteristics of the machine running the software; that is, IP address, MAC addresses, hostname, and so on.
- A cryptographic hash of one or more files on the filesystem.
- A cryptographic hash of the entire command line being executed by the program.

4.5 Usage

The following are the command-line options for **runwithpass**:

```
runwithpass.exe [ -conf <file> ] -res <resource ID> -acct <account ID>
  [ -expirecache ][ -keyword <string> ][ -replace <inputfile>
  <outputfile> ][ -v ][ -initial ][ -randomize ][ -override <password> ]
  [ -downloadfile <attributekey> ][ -downloaddir <directory> ]
  [ -downloadfilepassword ][-- <client cmd> ]
```

Table 4.1: runwithpass arguments

Argument	Description
-conf <*filename*>	Specify a configuration file. The default is **config.ini**.
-res <*resource ID*>	The ID of the system from which the password will be fetched.
-acct <*account ID*>	The ID of the account for which the password will be fetched.
-expirecache	Treat cached credential as expired. See Disabling caching in the *Reference Manual* for more information.
-keyword <*string*>	String to replace with password in client command arguments or input file.
-replace <*inputfile*> <*outputfile*>	Search/replace on the given input/output files. <*inputfile*> may be – meaning stdin. <*outputfile*> may be – meaning stdout.
-v	Attempts to obtain more detailed error information when available.
-initial	Set the initial passwords as encrypted. Using this argument will prompt for the following: • API user ID - The OTP IDAPI user • API user's password - The initial password for the OTP IDAPI credentials • Proxy user ID - The proxy user (if the `proxy` is configured in the configuration file) • Proxy user's password - The password of the proxy user • userkey - An unencrypted null-terminated string to add to the encryption key
-override <*password*>	Set the account's password to the specified value.
-randomize	Set the account's password to a random value.
-downloadfile <attributekey>	Download the large credential file using this attribute key.
-downloaddir <directory>	Download the large credential file to a specific directory. By default, the current directory will be used.
-downloadfilepassword	Fetch the password associated with the large credential file, if one exists. Must be used in conjunction with -downloadfile.

. . . continued on next page

Table 4.1: runwithpass arguments (Continued)

Argument	Description
--	Client command line and arguments to run follow the `--`. If the client command is omitted and no replacement is specified, the password is sent to stdout.

WARNING!: The command line, including the password if it was substituted, executed by **`runwithpass`** may be visible to other users of the system. Using -replace to pass the password to the program's standard input is recommended.

It is recommended that you use full path names in all arguments.

4.5.1 Examples

1. To fetch the password for psadmin on target system SSH:

```
runwithpass.exe -conf config.ini -res SSH -acct psadmin
```

2. To replace `PWD` in template.txt with the password for account APISVCACCT on system PAMSYSID01:

```
runwithpass -conf config.ini -keyword PWD -res PAMSYSID01 -acct APISVCACCT -
 replace template.txt - -- /bin/cat
```

3. To pass a password on the command-line to /usr/local/bin/somecommand:

```
runwithpass -conf config.ini -keyword PASSWORD -res PAMSYSID01 -acct APISVCACCT
 -- /usr/local/bin/somecommand -u APISVCACCT@myserver -p PASSWORD
```

   Note that running 'ps -ef' will display the password unless /usr/local/bin/somecommand removes it from its process space.

   There is no portable way for processes to hide their own command-line arguments and none at all for a parent process to rewrite command-line arguments after passing them to a child process.

4. To initialize the credential passwords with useargskey encryption:

```
runwithpass -res AD -acct Administrator -initial
```

5. To set the password of an account to a specified value:

```
runwithpass -res AD -acct Administrator -override newPassword
```

6. To randomize the password of an account:

```
runwithpass -res AD -acct Administrator -randomize
```

7. To download a large credential file to the current directory:

```
runwithpass -res TESTSYS -acct vaultfile -downloadfile LC_FILE
```


8. To download a large credential file to a specified directory:

```
runwithpass -res TESTSYS -acct vaultfile -downloadfile LC_FILE -downloaddir /home/
 psadmin
```

9. To download a large credential file that is associated with a password:

```
runwithpass -res TESTSYS -acct vaultfile -downloadfile LC_FILE -
 downloadfilepassword
```

See Chapter 38 in the Reference Manual for additional information on **`runwithpass`** and **`pamutil`**.

File Locations A

This chapter provides details of the location and purpose of files installed by:

- *Hitachi ID Bravura Security Fabric* (p26)
- *Hitachi ID Connector Pack* (p31)

When you install any Hitachi ID Systems product, the default path for program files is:

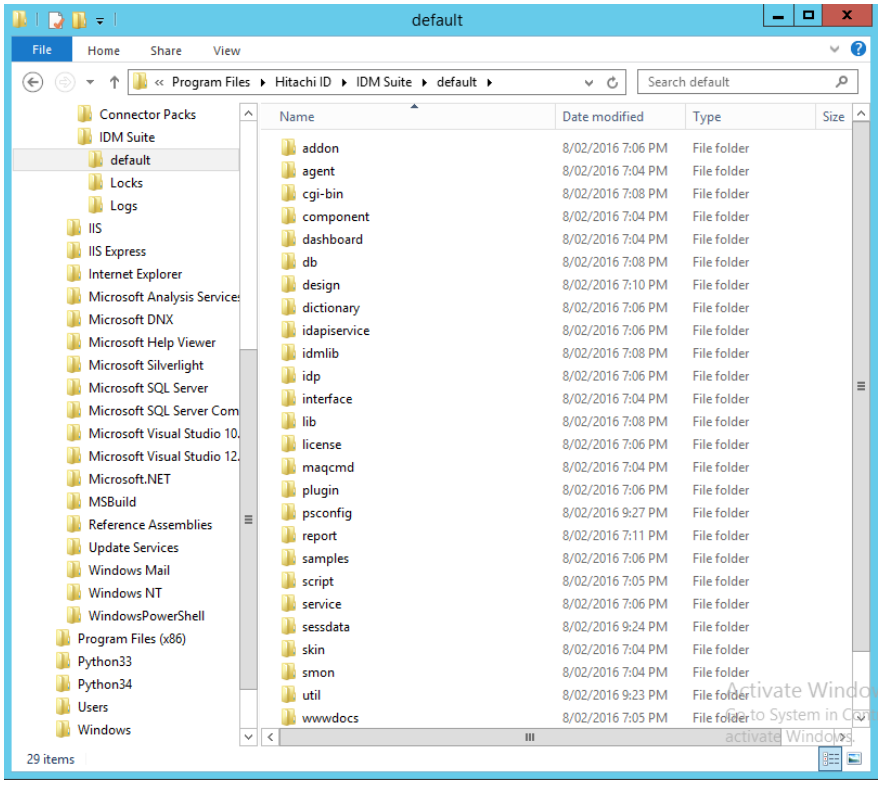
C:\Program Files\Hitachi ID\

A.1 *Bravura Security Fabric* directories and files

There are three main directories that are created when you install *Bravura Privilege* instance:

- <*Program Files path*>\Hitachi ID\IDM Suite\<*instance*>\
- <*Program Files path*>\Hitachi ID\IDM Suite\Logs\<*instance*>\
- <*Program Files path*>\Hitachi ID\IDM Suite\Locks\

The contents of those directories are detailed in the following subsections.



It is recommended that you do *not* change these directory locations during the setup process. You cannot install any of the directories required for *Bravura Privilege* on a mapped drive.

A.1.1 Instance directory

Instance directory files describes the function of directories that are created when an instance of *Bravura Privilege* is installed.

Note: Directories marked with ⋆ include files installed by *Connector Pack*.

Directories marked with ⋆⋆ include folders and files installed with the optional *Analytics* app.

Directories marked with † include optional files. They are only installed in a complete installation or if selected in a custom installation.

Table A.1: Instance directory files

	Directory	Contains
† ⋆	addon	Files required for add-on software. Some files, required to target Netegrity SiteMinder, are installed by *Connector Pack*. If you installed a global *Connector Pack*, these files are contained in the *Connector Pack* global directory.
⋆	agent	Instance-specific user management connectors (agents). If you installed a global *Connector Pack*, user management connectors are contained in the *Connector Pack* global directory.
⋆⋆	analytics	*Analytics* app specific folders
⋆⋆	analytics\DataSets	Contains **`*.rsd`** files which are Shared Dataset Definitions. These files are only used by SQL Server versions higher than Express. They contain datasets that are shared between reports.
⋆⋆	analytics\Hidden	Contains **`*.rdl`** files which are Report Definitions. These files are the drillthrough reports used by other reports. They are not visible to the end-user.
⋆⋆	analytics\ReportItems	This folder contains other folders. Each folder in this folder is a category in the *Analytics* app. Within these folders are **`*.rdl`** files which are Report Definitions. The folders need to be added to the **CUSTOM ANALYTIC CATEGORIES** system variable to be visible. These reports are then visible to the end-users in the *Analytics* app.
	cgi-bin	The user web interface modules (**`*.exe`** CGI programs).
	db	The *Bravura Privilege* database SQL scripts.
	db\cache	Search engine temporary search results. These files are cleaned up nightly by **`psupdate`**.
	db\replication	Stored procedure replication queues, and temporary replicated batch data.
⋆	design	Files necessary to make modifications to the GUI. Some files are installed by *Connector Pack*. If you install a global *Connector Pack*, files related to connectors are located in the global design directory. See the *Bravura Security Fabric* Documentation for details.
	dictionary	A flat file, **`words.dat`**, that contains dictionary words. *Bravura Privilege* uses this file to determine if new passwords fail dictionary-based password-policy rules.
	idapiservice	Files required to use the SOAP API.
⋆	interface	Instance-specific ticket management connectors (exit trap programs). If you installed a global *Connector Pack*, ticket management connectors are contained in the *Connector Pack* global directory.
	lib	Contains the **`pslangapi.dll`**.
	license	The license file for *Bravura Privilege*.
	plugin	Plugin programs executed by *Bravura Privilege*.
	psconfig	List files produced by auto discovery and the **`idmsetup.inf`** file.

. . . continued on next page

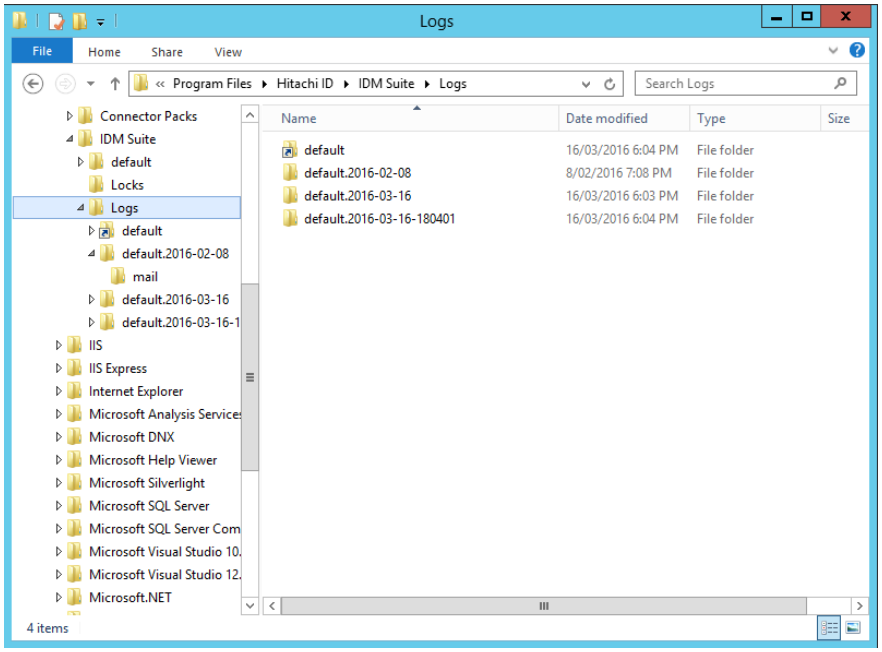
Table A.1: Instance directory files (Continued)

	Directory	Contains
	report	Files and programs for report generation.
† ⋆	samples	Instance-specific sample scripts and configuration files. If you installed a global *Connector Pack*, connector-related sample files are contained in the *Connector Pack* global directory.
	script	Configuration files and scripts used by connectors, **`psupdate`**, plugins and interface programs.
	service	Service programs.
	sessdata	Session data. A scheduled program removed old data files nightly.
	skin	Compiled GUI files used at run-time (HTML and *.z).
	smon	Monitored session data. This location can be changed by *Recorded session management* (SMON) module options.
⋆	util	Command-line programs and utilities. If you install a global *Connector Pack*, tools related to connector configuration are located in the global util directory.
⋆	unix	The **`psunix`** archive, which is required to install the Unix Listener and supporting files on a Unix-based target system. If you installed a global *Connector Pack*, this directory is created in the *Connector Pack* global directory.
	wwwdocs	Images and static HTML pages used by *Bravura Privilege*.

A.1.2 Log directory

Any operation that is run by *Bravura Privilege* is logged. Those logs are invaluable when debugging an issue. The log directory by default is C:\Program Files\Hitachi ID\IDM Suite\Logs\. Each instance of *Bravura Privilege* that is installed will have at least one sub-directory within this directory.

The rotatelog scheduled job, which runs on a nightly basis, rotates the logs in to a new folder, to reduce disk space usage.



See the *Bravura Security Fabric* Documentation for more information.

A.1.3 Locks directory

Certain target systems can only be accessed serially, such as Lotus Notes. This is a limitation of the API used to access the target system. In these cases *Bravura Privilege* drops a *lock file* in the locks directory when an operation is being performed that should only be performed serially. For this reason the locks directory *must* be the same for all instances of *Bravura Privilege* that are installed on the same server.

See the *Bravura Security Fabric* Documentation for more information.

A.2 Connector pack directories and files

When you install *Hitachi ID Connector Pack*, files are placed in different locations depending on type of *Connector Pack*.

For an instance-specific connector pack, the installer, **`connector-pack-x64.msi`**, installs connectors and supporting files in:

<*Program Files path*>\Hitachi ID\IDM Suite\<*instance*>\

For a global connector pack, the installer, **`connector-pack-x64.msi`**, installs connectors and supporting files in:

<*Program Files path*>\Hitachi ID\Connector Packs\global\

Connector Pack directory files describes the function of directories that are created when a *Connector Pack* is installed:

Table A.2: Connector Pack directory files

Directory	Contains
addon	Files required to target Netegrity SiteMinder systems
agent	User management connectors (agents)
design	*Connector Pack*-related files necessary to make modifications to the GUI; for example target system address help pages. See the *Bravura Security Fabric* Documentation for details.
interface	Ticket management connectors (exit trap programs)
samples	Sample scripts and configuration files
unix	The **`psunix`** archive, which is required to install the Unix Listener and supporting files on a Unix-based target system
util	Tools to support the configuration of various target systems