APPLICATION ACCESS MANAGER

AAM INTEGRATION - TECHNICAL DOCUMENTATION TEMPLATE

DOCUMENT PURPOSE: THIS TEMPLATE IS TO BE COMPLETED BY PARTNER AND IS REQUIRED FOR CYBERARK SECURED CERTIFICATION. THE AAM INTEGRATION TECHNICAL DOCUMENTATION WILL BE MADE AVAILABLE TO PARTNERS, CUSTOMERS AND PROSPECTS.

Name of Company	Celonis
Website	https://www.celonis.com/de/
Name of Product	Celonis Process Mining
Version	4.7.1 and higher
Date	JUNE 16, 2021

PARTNER SOLUTION OVERVIEW

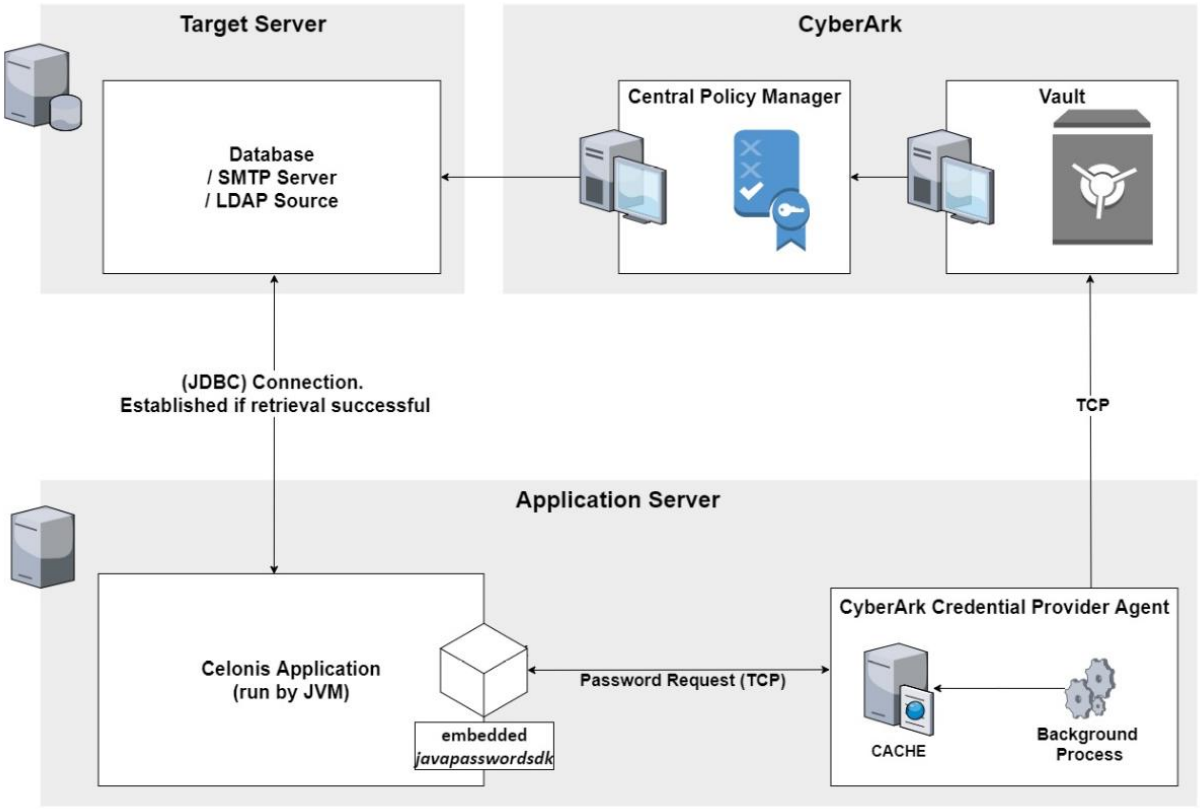
Celonis is a powerful software for retrieving, visualizing, and analyzing real as-is business processes from transactional data. It provides users with the possibility to create and share comprehensive process analyses giving them full transparency about the business processes at hand.
The software consists of a web application that can connect to various databases. The installation package is available as Windows Installer or a Linux executable run file for common distributions (RHEL7, SLES12, Ubuntu18). The integration will be implemented for application version 4.7.1 and higher.

KEY BENEFITS

Celonis is an enterprise-proven Process Mining solution that analyzes all kinds of processes. It offers powerful features for the administration of user roles, data, and analyses. It effortlessly scales for even the largest Big Data scenarios while maintaining a low data footprint.

With this integration, the Celonis end-user does not have to enter sensitive database credentials into the configuration files or the frontend of the application anymore.

PRODUCT DIAGRAM & DESCRIPTION OF PRODUCT INTEGRATION



Every instance of the *Application* service (*Application#1, Application#N*):

- Is independent (not depending on load-balancers, firewalls, common configuration files, etc.)
- Has its own SDK classes
- Has its own configuration files/properties
- Is *Agent AAM Based* hence "requires a dedicated AAM agent to be locally installed" (refer to *Application Access Manager Integration Instructions in the AAM INTEGRATION* GUIDE). This means each *Application#N* requires that the service is running on the same instance (**`aimprv`** for Linux, **`CyberArk Application Password Provider`** service for Windows)
- Requires the registration of hostnames for each *Application#N* to access the Safe/Vaults (refer to *Application Access Manager Integration Instructions - Testing the Installation*

Considering these points, the customer is free to install and run as many instances of the application as needed. No other mid-tier, load-balancers, firewalls, or endpoints configuration is required.

Triggering the credential retrieval:

> **Configuration Files:** The actual values are retrieved from CyberArk as soon as the request is resolved by the Java Spring application. In the current implementation this happens during the Spring Context initialization phase when the application is started.
>
> **Frontend/UI configuration: The actual values are retrieved from CyberArk only upon creation or update of the established data connection (see #*Applicable passwords* section below for the complete list of configurable CyberArk** connection settings).

Circumstances and frequency of the retrieval:

> **Configuration Files:** Usually, the application service is stopped for configuration and updates. The frequency depends on the size, landscape and use cases of the customer.
>
> **Frontend/UI configuration:** This highly depends on the state of the customers implementation. Any frequency from once per week to multiple times per day is realistic.

AAM INSTALLATION

Refer to the "Credential Provider and ASCP Implementation Guide" or the "Central Credential Provider Implementation guide" for CyberArk Agent based or Agentless installation and configuration.

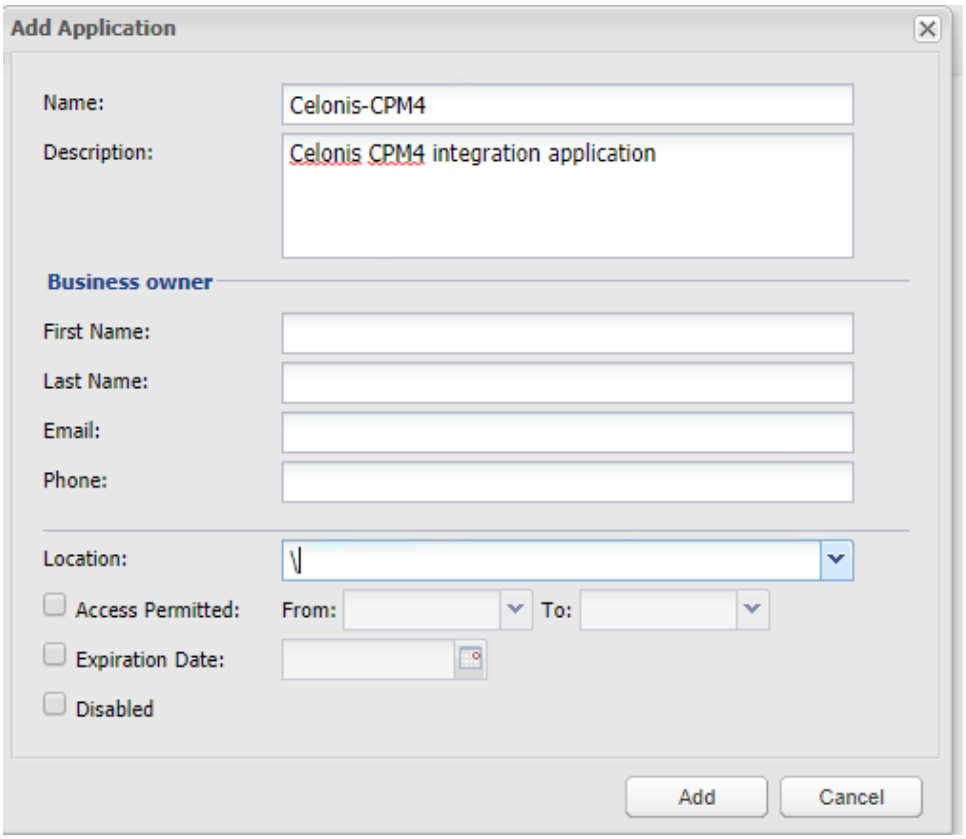
Integrating CPM4 with the CyberArk AAM Agent based installation requires the customer to place the *javapasswordsdk.jar* runtime library in the *<installDir>/lib* path of the Celonis 4 installation directory. The jar file can be found in the CyberArk service installation directory (typically, */opt/CARKaim/sdk/javapasswordsdk.jar* for Linux, *C:\Program Files (x86)\CyberArk\ApplicationPasswordSdk\JavaPasswordSDK.jar* for Windows).

AAM CONFIGURATION

DEFINING THE APPLICATION ID (APPID) AND AUTHENTICATION DETAILS

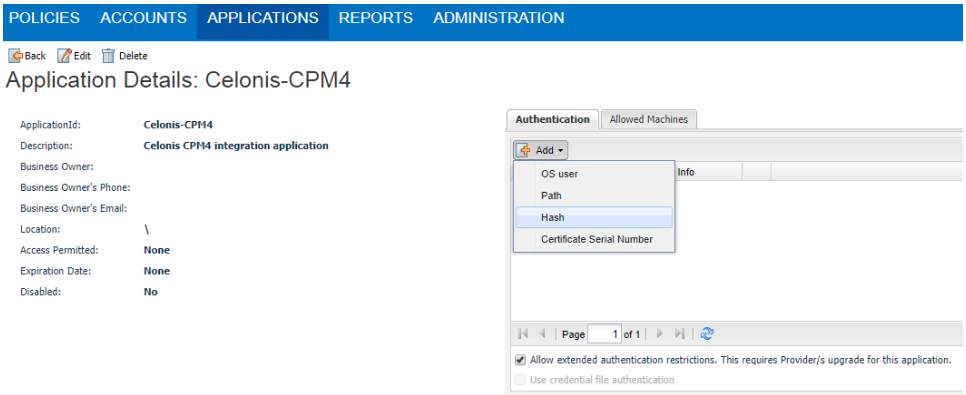
To define the Application, here are the instructions to define it manually via CyberArk's PVWA (Password Vault Web Access) Interface:

1. Logged in as user allowed to managed applications (it requires Manage Users authorization), in the Applications tab, click **Add Application**; the Add Application page appears.



2. Specify the following information:
    - In the Name edit box, specify the unique name (ID) of the application. The recommended Application ID for this integration is: **APP ID = Celonis-CPM4**
    - In the Description, specify a short description of the application that will help you identify it.
    - In the Business owner section, specify contact information about the application's Business owner (see ***PARTNER CONTACT INFO*** section or request the actual Celonis business contact).
    - In the most cases root location **/** is sufficient for CPM4 integration

3. Click **Add**; the application is added and is displayed in the Application Details page.



- **Allowing extended authentication restrictions**. This enables you to specify an unlimited number of machines and Windows domain OS users for a single application. Please check this box.

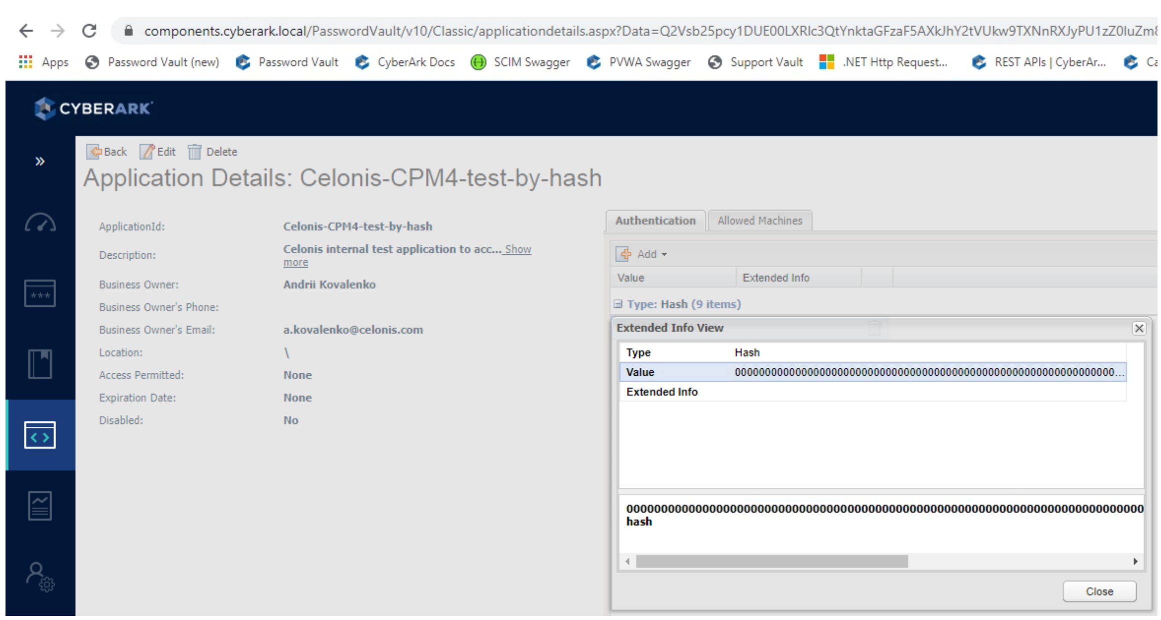
4. Specify the application's **Authentication** details. This information enables the Credential Provider to check certain application characteristics before retrieving the application password. Partner should suggest the authentication details to secure the access by the partner's application.

   It is recommended using the hash authentication method. The full guide on how to create an application hash value can be found here:
   https://docs.cyberark.com/Product-Doc/OnlineHelp/AAM-CP/Latest/en/Content/CP%20and%20ASCP/Generating-Application-Hash-Value.htm

   Specifying the application hash:
   Caveat: It might take some time to for the configuration to apply after submitting the hash values within the CyberArk configuration.

   The CPM4 application is a complex application, which includes classes, wrapped to several levels of `*.jar` and `*.war` files, running inside the web-application service, which is executed by a `jsvc` runner. Therefore, calculating a HASH of the specific class using the **`AIMGetAppInfo`** utility is not possible under every circumstance. Thus, the following workaround is recommended to calculate the hashes:

   a. By default, the application hash is not validated. "Enable" the hash authentication at *Components->Application->Hash* configuration by specifying a dummy value (for instance:
   `000000000000000000000000000000000000000000000000000000000000000000000000000000000000000`
   `0000000000000000000000000000000000000000000#dummy hash`
   ):



   You can add additional information in a comment after each hash value specified for an application by putting '#' after the hash value, followed by the comment.

   b. Start the CPM4 application (`sudo ./start.sh` or starting the Windows Service)

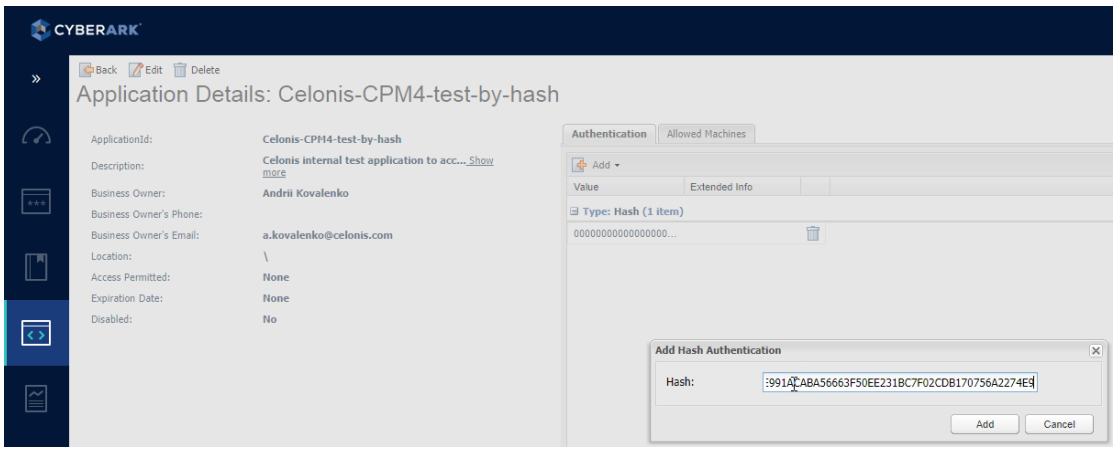
c. Review the *error* logs (by default - `logs/stderr`) and find the CyberArk exception message regarding the application hash authorization:

```
$ sudo tail -n 100 -f logs/stderr
…
de.celonis.pm.utils.properties.exception.PropertyResolveException: Can't
retrieve password for Cyberark property "appid=Celonis-CPM4-test-by-
hash&safe=Celonis-CPM4-Safe&object=database.password&reason=cpm4-
application-local-test".
Reason: APPAP133E Failed to verify application authentication data: Hash
"5C4B13E3D22EC9DD00EBBAAE352FDAA1DF5D83A6B9D52FD76D586BE163AE8AD13BAB52BCB
B907B64E81BE991ACABA56663F50EE231BC7F02CDB170756A2274E9" is unauthorized
Caused by: class javapasswordsdk.exceptions.PSDKException: APPAP133E
Failed to verify application authentication data: Hash
"5C4B13E3D22EC9DD00EBBAAE352FDAA1DF5D83A6B9D52FD76D586BE163AE8AD13BAB52BCB
B907B64E81BE991ACABA56663F50EE231BC7F02CDB170756A2274E9" is unauthorized
```

d. The same result could be found in the **`aimprv`** service error logs:

```
$ sudo service aimprv status
Jun 07 11:34:26 CEL-LP-961 appprovider[23298]: CyberArk AIM[23298]:
APPAU002E Provider [Prov_CEL-LP-961] has failed to fetch password
with query [Object=database.password;Safe=Celonis-CPM4-Safe] for
application [Celonis-CPM4-test-by-hash]. Fetch reason: [APPAP133E
Failed to verify application authentication data: Hash
"5C4B13E3D22EC9DD00EBBAAE352FDAA1DF5D83A6B9D52FD76D586BE163AE8AD13B
AB52BCBB907B64E81BE991ACABA56663F50EE231BC7F02CDB170756A2274E9" is
unauthorized]
```

e. Add this hash to *Components->Application->Hash*:



f. Restart the CPM4 application: `sudo ./stop.sh ; sudo ./start.sh` or by starting and stopping the Windows service

g. Review the *stdout* logs (`logs/stdout`) or the aimprv service logs (`sudo service aimprv status`) to find the new full-path class hash, including all `*.jar`/`*.war` wrappers:

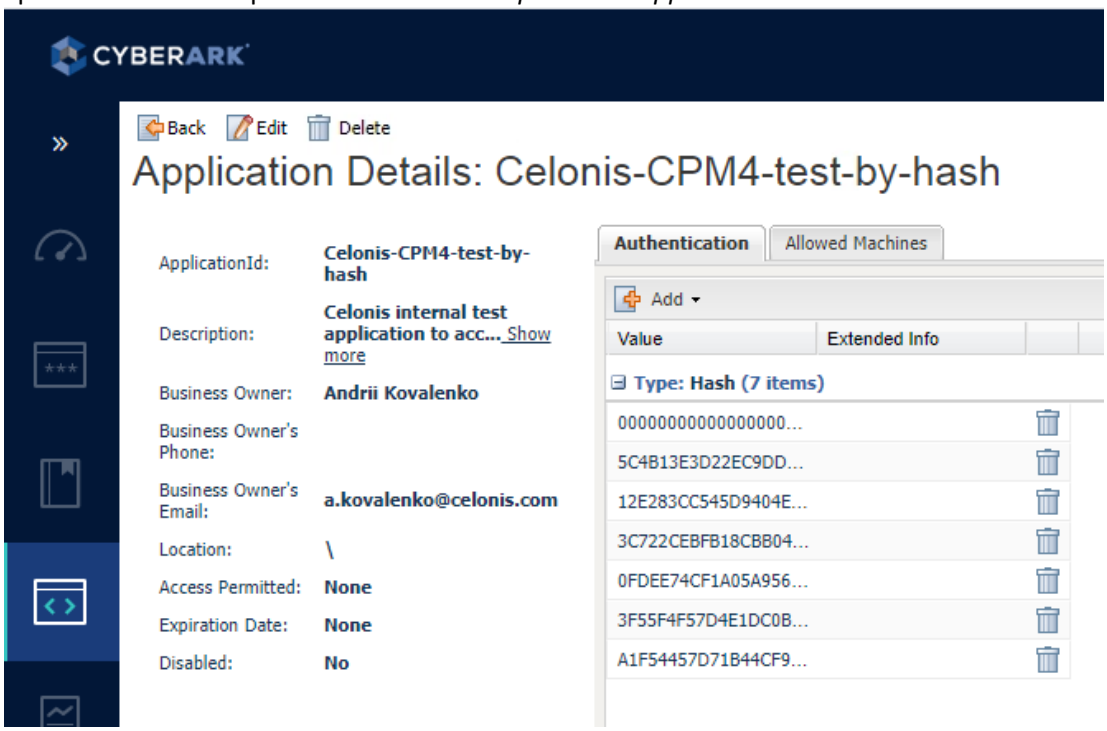
```
$ sudo tail -n 100 -f logs/stdout
Can't retrieve password for Cyberark property "appid=Celonis-CPM4-test-by-
hash&safe=Celonis-CPM4-Safe&object=database.password&reason=cpm4-
application-local-test".
Reason: APPAP133E Failed to verify application authentication data: Hash
"5C4B13E3D22EC9DD00EBBAAE352FDAA1DF5D83A6B9D52FD76D586BE163AE8AD13BAB52BCB
B907B64E81BE991ACABA56663F50EE231BC7F02CDB170756A2274E9;12E283CC545D9404E3
571AC4DDA80E1A27644CF254C37A3FC164F543D57125C68B2FCBCE8C4685216AEAB80A23C6
```

```
E05484C394C275339CEAC3C03448E6496B22;3C722CEBFB18CBB049BC55DF6707AEC497952
D8F1ECDC7EEB876613013A299DA212EC812361BCC6CA32FC7FB5E5278E298B588902F42FF5
19DF514C3C27F17BD;0FDEE74CF1A05A9562EE8059FF33D4450E07FC4A94C31F5BE449235B
D34E45030FA29BEA0FF1929C7D5052577A5DD5794A2C65956783BBB207C4E53584C868D5;3
F55F4F57D4E1DC0B30110E52D379E536ADA0B613FDBC905661FAEF5CB022BAAFD0F4D41818
59974911A753C73820A134D77553C696C4C3E075F261BD4D3DF8B;A1F54457D71B44CF9069
58F0552951F753B3B7BEB05CC5660D913388F0910ED72D0C302C5150EC49641DA174E46C07
B52FE75D8DAF0DE1D2130212089E1B46D7" is unauthorized; nested exception is
de.celonis.pm.utils.properties.exception.PropertyResolveException: Can't
retrieve password for Cyberark property "appid=Celonis-CPM4-test-by-
hash&safe=Celonis-CPM4-Safe&object=database.password&reason=cpm4-
application-local-test".

Caused by: class javapasswordsdk.exceptions.PSDKException: APPAP133E
Failed to verify application authentication data: Hash
"5C4B13E3D22EC9DD00EBBAAE352FDAA1DF5D83A6B9D52FD76D586BE163AE8AD13BAB52BCB
B907B64E81BE991ACABA56663F50EE231BC7F02CDB170756A2274E9;12E283CC545D9404E3
571AC4DDA80E1A27644CF254C37A3FC164F543D57125C68B2FCBCE8C4685216AEAB80A23C6
E05484C394C275339CEAC3C03448E6496B22;3C722CEBFB18CBB049BC55DF6707AEC497952
D8F1ECDC7EEB876613013A299DA212EC812361BCC6CA32FC7FB5E5278E298B588902F42FF5
19DF514C3C27F17BD;0FDEE74CF1A05A9562EE8059FF33D4450E07FC4A94C31F5BE449235B
D34E45030FA29BEA0FF1929C7D5052577A5DD5794A2C65956783BBB207C4E53584C868D5;3
F55F4F57D4E1DC0B30110E52D379E536ADA0B613FDBC905661FAEF5CB022BAAFD0F4D41818
59974911A753C73820A134D77553C696C4C3E075F261BD4D3DF8B;A1F54457D71B44CF9069
58F0552951F753B3B7BEB05CC5660D913388F0910ED72D0C302C5150EC49641DA174E46C07
B52FE75D8DAF0DE1D2130212089E1B46D7" is unauthorized
```

NOTE: There are multiple hashes that are semicolon separated

h. Split the hashes and put all of them to *Components->Application->Hash*:



i. Restart CPM4 application: `sudo ./stop.sh ; sudo ./start.sh` or by starting and stopping the Windows service.

j. **The hash values are now specified and ready to be used.**

PROVISIONING ACCOUNTS AND SETTING PERMISSIONS FOR APPLICATION ACCESS

For the application to perform its functionality or tasks, the application must have access to existing accounts, or new accounts to be provisioned in CyberArk Vault (Step 1). Once the accounts are managed by CyberArk, make sure to setup the access to both the application and CyberArk Application Password Providers serving the Application (Step 2).

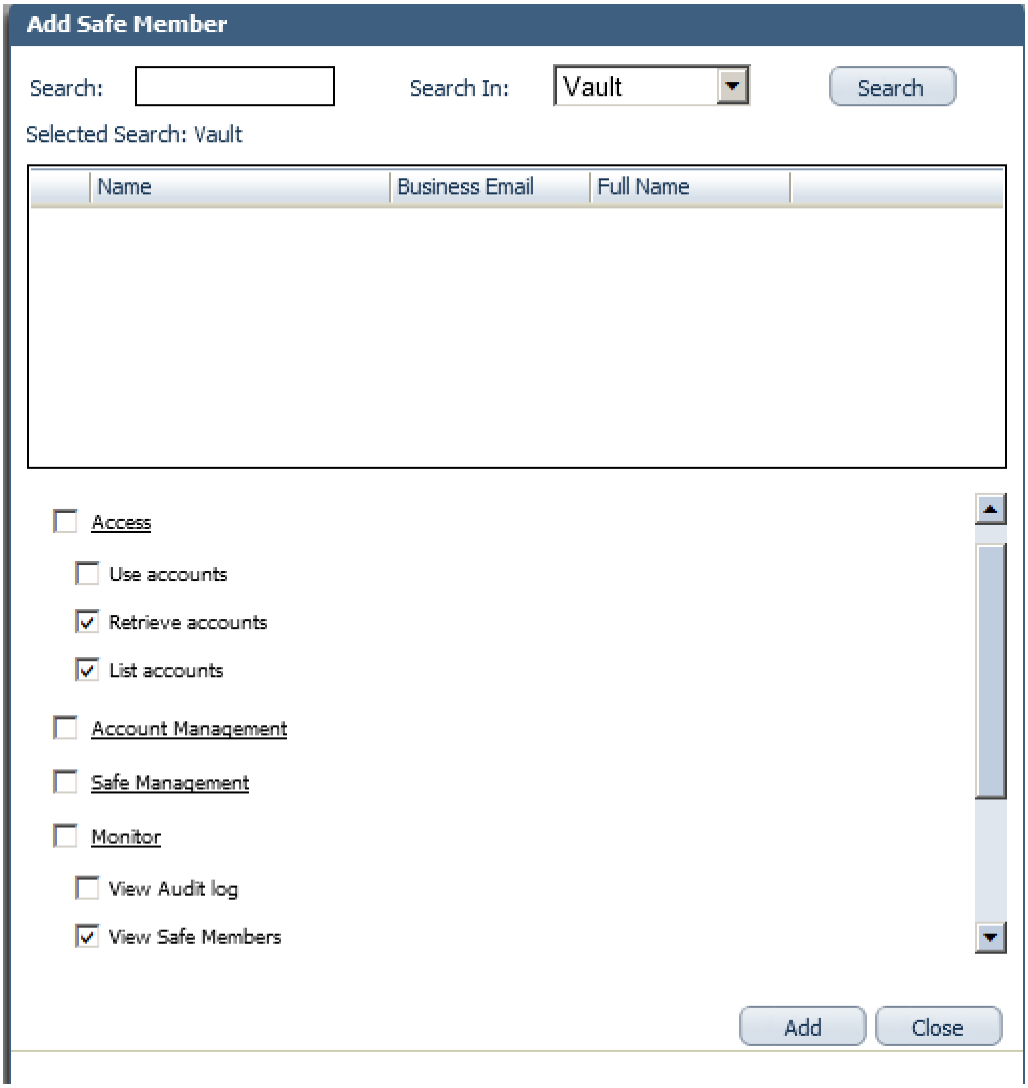
1. In the Password Safe, provision the privileged accounts that will be required by the application. You can do this in either of the following ways:
   - **Manually** – Add accounts manually one at a time and specify all the account details.
   - **Automatically** – Add multiple accounts automatically using the Password Upload feature.

     For this step, you require the **Add accounts** authorization in the Password Safe.

   For more information about adding and managing privileged accounts, refer to **the Privileged Access Security Implementation Guide.**

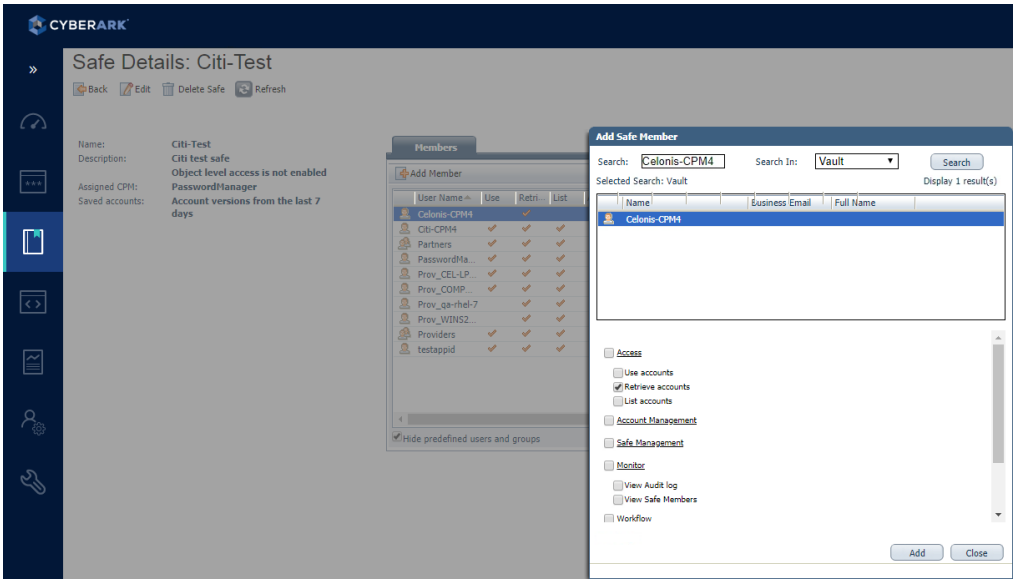
2. Add the Credential Provider and application users as members of the Password Safes where the application passwords are stored. This can either be done manually in the Safes tab, or by specifying the Safe names in the CSV file for adding multiple applications.
   i. Add the Provider user as a Safe Member with the following authorizations:
      - List accounts
      - Retrieve accounts
      - View Safe Members

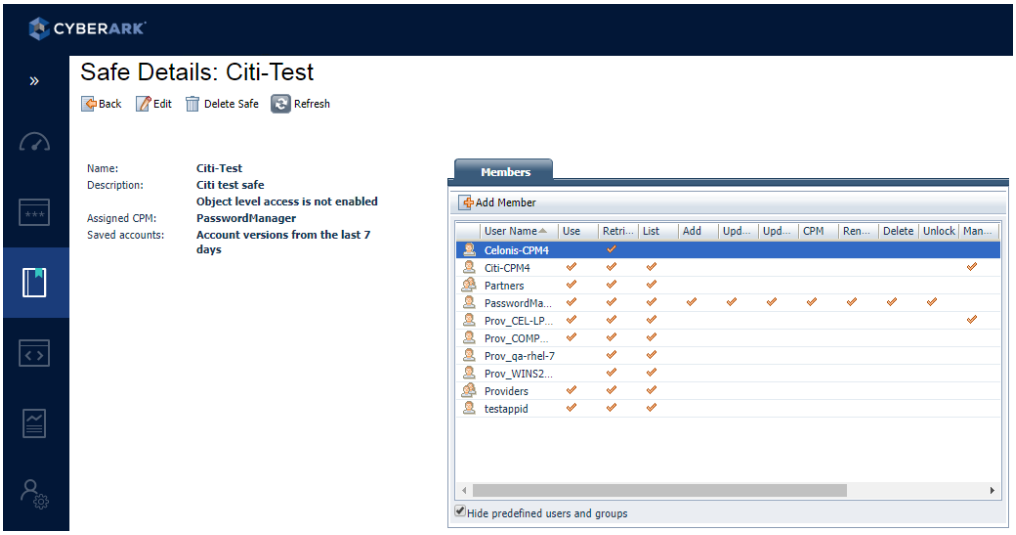
        Note: When installing multiple Providers for this integration, it is recommended to create a group for them, and add the group to the Safe once with the above authorization.



ii. Add the application (the APPID) as a Safe Member with the following authorizations:

- Retrieve accounts





iii. If your environment is configured for dual control:

- In PIM-PSM environments (v7.2 and lower), if the Safe is configured to require confirmation from authorized users before passwords can be retrieved, give the Provider user and the application the following permission:
    - Access Safe without Confirmation
- In Privileged Account Security solutions (v8.0 and higher), when working with dual control, the Provider user can always access without confirmation, thus, it is not necessary to set this permission.

iv. If the Safe is configured for object level access, make sure that both the Provider user and the application have access to the password(s) to retrieve.

For more information about configuring Safe Members, refer to the **Privileged Access Security Implementation Guide.**

PARTNER PRODUCT INSTALLATION & INTEGRATION CONFIGURATION

Refer to the **Celonis Installation Guide (Celonis Operation Guide)** for Partner Product installation. You will find a separate section on how to set up the integration there.

Password retrieval
After connecting Celonis to CyberArk, the Java Properties of every custom ***. properties** file inside the Celonis installation directory can be configured for retrieval via CyberArk.

Prerequisites

1. The **`javapasswordsdk.jar`** runtime library supplied by CyberArk has been placed in *the <installDir>/*lib folder in the Celonis 4 installation directory.
2. The CyberArk Credential Provider Agent (**`aimprv`** service on Linux, **`CyberArk Application Password Provider`** Service on Windows) is running on the same instance as the Celonis service.

Properties configuration
The properties to be retrieved via CyberArk need to have the following format:
<<property.name>>=**cyberark-sdk:<<**LIST_OF_OBJECT_ARGUMENTS>>

With:

<<property.name>>	Java Property name to be retrieved. For example *database.password.*
cyberark-sdk:	Mandatory prefix for the use of CyberArk (colon included)
<<LIST_OF_OBJECT_ARGUMENTS>>	URL-encoded string of CyberArk object request arguments (e.g. AppID, Safe, Object, Reason) in a URL query format. Properties are separated by "**&**". Property name and value are separated by "**=**".

Example

*database.password=**cyberark-sdk:**appid=yourcompanyappid&safe=safename&object=objectname&reason=cpm4-application-db-configuration*

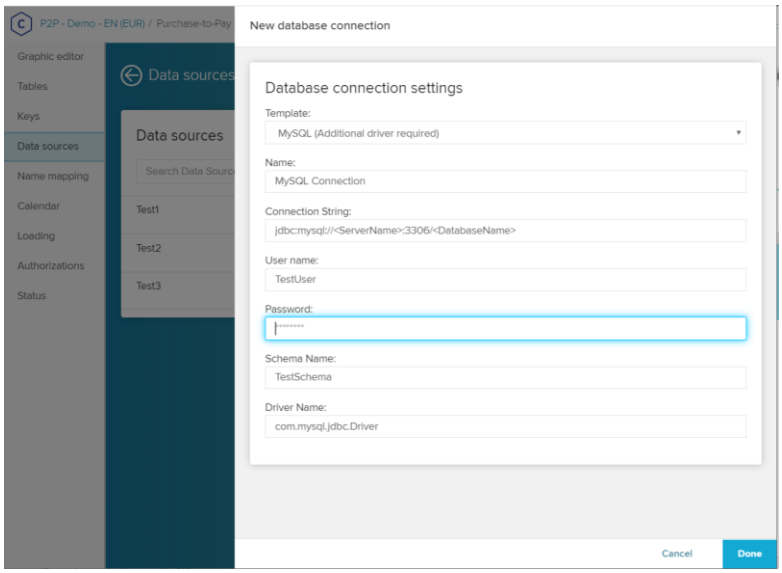
Frontend configuration
The frontend configuration follows the same rules as the configuration of the properties. Retrieving the passwords requires the following format:

cyberark-sdk:<<LIST_OF_OBJECT_ARGUMENTS>>

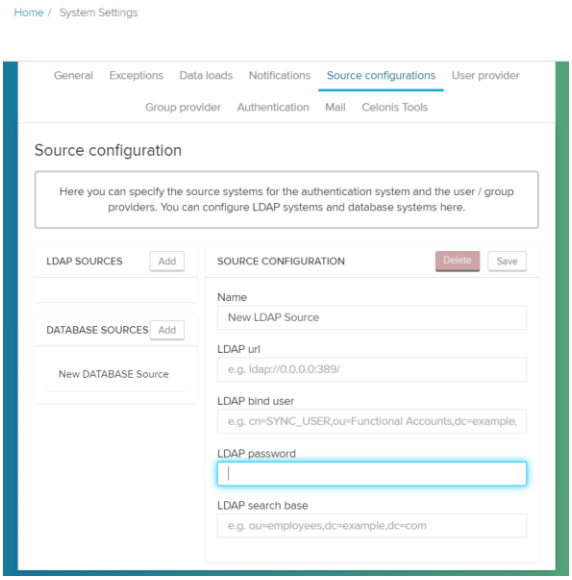
Example
cyberark-sdk:*appid=yourcompanyappid&safe=safename&object=objectname&reason=cpm4-application-db-configuration*

Applicable passwords
The frontend integration is limited to four specific fields. The following password fields can be retrieved from CyberArk:

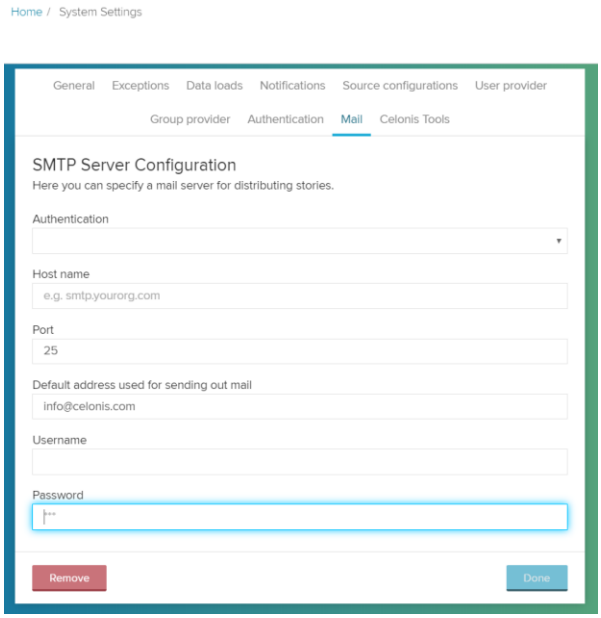
- The password to connect to a database from within a Data Model:



- The "LDAP password" in *System Settings → Source Configurations → LDAP Sources* and the "Database password" in *System Settings → Source Configurations → Database Sources:*



- The SMTP Server Password *in System Settings → Mail*



Notes

- **appid, safe, object** and **reason** are typical CyberArk request arguments. This example could be extended according to all **single String setter names** (e.g. **setPolicyID(String) -> policyid, setFolder(String) -> folder, ...**) that are supported by the CyberArk Java SDK. Please follow the PSDKPasswordRequest java class documentation for all supported arguments.
- The request arguments are case-insensitive
- As <<LIST_OF_OBJECT_ARGUMENTS>> is a URL-encoded string, one could leverage the usage by URL-encoding the values. For example the request with reason="Some reason" and extended chars: []{}\\/ [陰]{陽}" could look like this:

*database.password=**cyberark-sdk:**appid=testappid&safe=test&object=cpm4&reason=%22Some%20weird%20quoted%20reasn'%20with%20extended%20chars%3A%20%5B%5D%7B%7D%2F%2C%20and%20chinese%20hieroglyphs%20%5B%E9%99%B0%5D%7B%E9%99%BD%7D%22*

Celonis recommends the usage of the AAM integration for the following exemplary cases:

- In **Configuration Files** (`config-custom.properties`, `component_configurations/access-logging.properties`, `component_configurations/login-logging.properties`) to protect the following properties (if used by the customer):
    - **`database.password`**
    - **`mail.password`** if active
    - **`server.ssl.*`** properties if SSL connection to the running CPM4 instance is enabled
    - **`saml.keystore.*`** properties, if SAML authentication is enabled
    - **`jwt.secret`**
    - **`access_logging.database_password`**, if access logging to the database is configured (access_logging.enabled=true && access_logging.database=true)
    - **`login_logging.database.password`**, if login logging to the database is configured (login_logging.enabled=true && login_logging.database=true)
    - any other properties which the customer considers to be sensitive.

- In **Frontend/UI**:
    - Depending on how strongly the customer trusts the security of the CPM4 application configuration database, Celonis recommends:
        - If the customer trusts the database protection – additional protecting of the Frontend/UI properties is not needed
        - If the customer expects that data can leak from the configuration database – the customer could apply additional protection to the passwords described in ***Applicable passwords*** above (Data Models, LDAP/Database sources, SMTP).

PARTNER CONTACT INFO

Business Contact	Name	Matthias Höfler *Partner Manager*
	Email	m.hoefler@celonis.com
	Tel	+13 4746 13566
Technical Contact	Name	Andrii Kovalenko *Engineering Lead*
	Email	a.kovalenko@celonis.com
	Tel	+49 (1516) 8943150
Support Contact	Name	Jörg Dörries *Head of Customer Support*
	Email	j.doerries@celonis.com
	Tel	+49 8921 5396144