OPERATION GUIDE

Version 1.16

Corresponding Software Version
Celonis 4.7.3.2

TABLE OF CONTENTS

REVISION HISTORY

VERSION NUMBER	VERSION DATE	SUMMARY OF REVISIONS MADE
1.4	MAR 22, 2017	Application Version 4.2
1.6	FEB 23, 2018	Updated version for application version 4.3
1.7	MAI 12, 2018	Updated version for application version 4.4
1.9	MAR 31, 2019	Updated version for application version 4.5
1.10	DEC 03, 2019	Updated version for application version 4.6
1.11	MAY 12, 2020	Updated version for application version 4.6.1
1.12	JAN 23, 2021	Updated version for application version 4.7
1.13	JUN 21, 2021	Updated version for application version 4.7.1
1.14	OCT 23, 2021	Updated version for application version 4.7.2
1.15	MAY 20, 2022	Updated version for application version 4.7.3
1.16	JAN 02, 2023	Updated version for application version 4.7.3.2

INTRODUCTION

ABOUT THIS GUIDE

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.

TARGET AUDIENCE

This guide covers all relevant technical information about correctly and securely operating and configuring Celonis and is meant to be consulted by the following target audiences:

- System Administrators
- Support Personnel
- Technical Staff

LIST OF ABBREVIATIONS

ABBREVIATION	EXPLANATION
AES	Advanced Encryption Standard
C	C – Imperative Computer Programming Language
CA	Certification Authority
CPU	Central Processing Unit
CRT	Certificate
CSR	Certificate Signing Request
CSV	Character-Separated Values
DB	Database
ERP	Enterprise Resource Planning
EXE	Executable (common filename extension)
GB	Gigabyte
HA	High Availability
HTTP	Hypertext Transfer Protocol
HSQLDB	Hyper SQL Database
ID	Identifier
JDBC	Java Database Connectivity
JKS	Java KeyStore
JMX	Java Management Extensions
JRE	Java Runtime Environment
OS	Operating System
PDF	Portable Document Format
PID	Process Identification Number

RAM	Read Access Memory
RSA	RSA public key cryptography algorithm
SAML	Security Assertion Markup Language
SHA	Secure Hash Algorithm
SQL	Structured Query Language
SSL	Secure Sockets Layer
URL	Uniform Resource Locator
VM	Virtual Machine
XLS	Excel Spreadsheet
ZIP	Zipper (Archive File Format)

TECHNICAL CONFIGURATION – SYSTEM LANDSCAPE

This section gives an overview of the Celonis architecture as well as its involvement with other systems in the IT landscape.

Celonis consists of two components, namely the core Process Mining Central Application and the Compute Service which serves for holding the data of loaded Data Models. To show you how the Celonis software works, we are presenting the Celonis System Landscape as a diagram below. The System Landscape itself contains sufficient information and visual elements so that you can fully understand how Celonis works and connects with your existing IT Infrastructure.

Celonis recommends the single-server deployment for nearly all use cases.

SINGLE-SERVER DEPLOYMENT

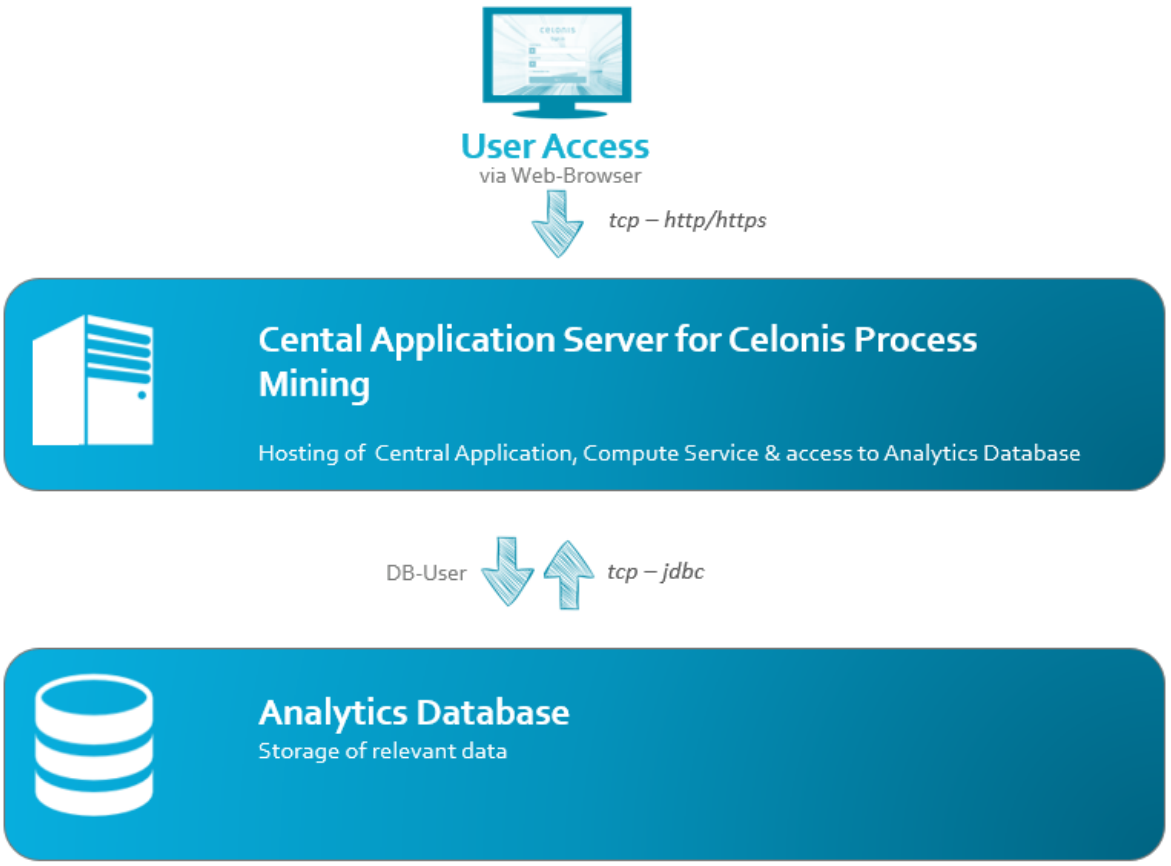

Figure 1: Single-server deployment

MULTI-SERVER DEPLOYMENT (SCALE-OUT)

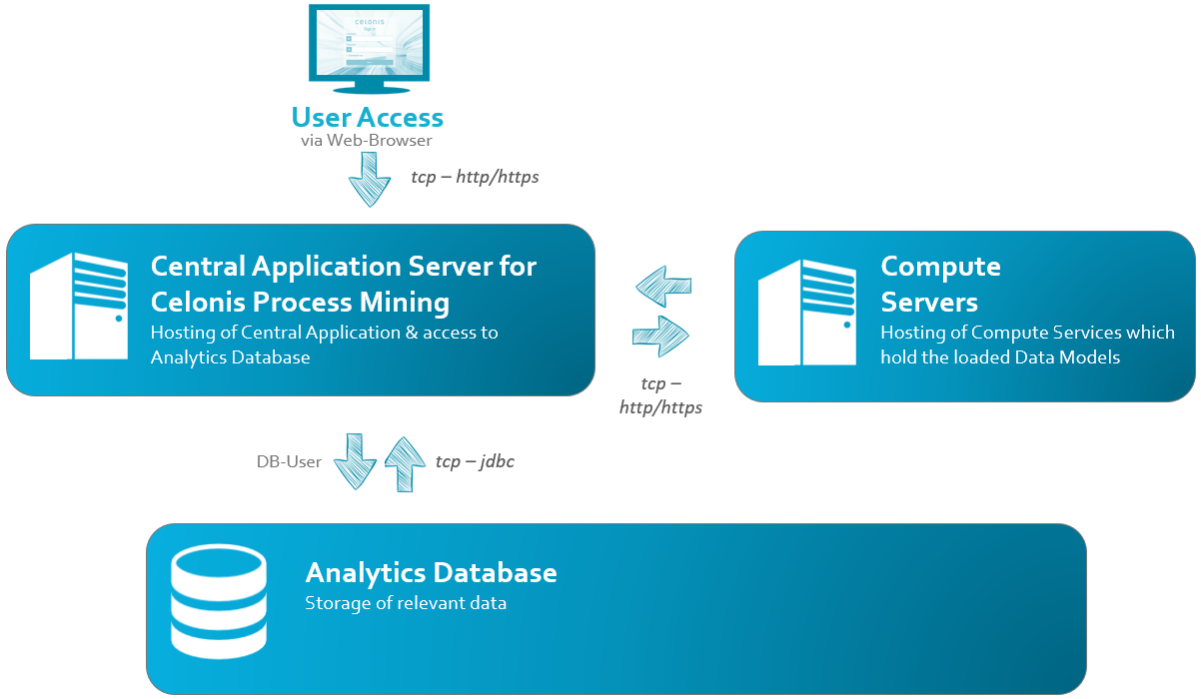

Figure 2: Multi-server deployment

TECHNICAL CONFIGURATION – FILE SYSTEM LAYOUT

This section describes the Celonis files and folders structure in detail, with a highlight on available configuration files.

The Celonis file system layout simplifies the process of understanding, managing and administering the application. The entire layout is split in three main sections: The application installation path , the application data path, where the application will create its files, and the compute data path, where the Compute Service will create files. All locations can be changed during the installation process. For the Windows Operating Systems, when kept to their default values, Celonis will install under “C:\Program Files\Celonis 4 Enterprise” and the Compute files will be created under “C:\Program Files\Celonis 4 Enterprise\compute”. The respective default values for Linux systems are “/opt/celonis/cpm4/” and “/opt/celonis/cpm4/compute”. Next, we will list and describe the folder tree for both Windows and Linux systems.

WINDOWS SYSTEMS

- "appfiles": Directory – Contains all the Celonis generated application files. From the Operations Guide perspective, the following files are relevant:
  - "uploads" – containing all files uploaded into Celonis (images, transports, .XLS, .CSV)
  - "backup" – containing all the configuration store backup snapshots.
- "component_configurations": Directory – Contains specific component configuration files (in a new installation, only sample files) that can be used to address specific component settings for access, audit, login and trace logging, password rules and query-definitions.
- "jre": Directory – Contains the AdoptOpenJDK JRE embedded package.
- "lib": Directory – Contains **custom** libraries and drivers.
- "logs": Directory – Contains the Celonis application log files.
- "logging.xml" - File that contains the default logging configuration
- "pdf-exporter": Directory – Contains the PDF Exporting capabilities functions.
- "temp": Directory – Contains temporary application files.
- ".cpm.sys.vmoptions": File – Contains system configuration of the JVM.

  DO NOT EDIT this file, as it will get overwritten in case of a product upgrade. Use the "cpm.user.vmoptions" file instead.
- ".last-version.run": File - contains last launched version of CPM
- "config.properties": File – Celonis configuration file that contains all the installation parameters.

  DO NOT EDIT this file, as it will get overwritten in case of a product upgrade. Use the "config-custom.properties" file instead.
- "config-custom.properties.sample": File – Sample Celonis configuration file that can be used as a starting point to change the Celonis Application Server configuration settings.
  - Parameters related to the Celonis Application Server listening interface, port, SSL, logging, SAML or multi-server deployment can be defined here.
  - Copy and rename this file to "config-custom.properties" to configure custom values.
  - A change of parameters in this file requires a restart of the application to take effect.
- "cpm.jar": File – Main Celonis Frontend executable file.
- "cpm.user.vmoptions.sample": File – Sample JVM configuration file that can be used as a starting point to insert custom JVM parameters.
  - Copy and rename this file to "cpm.user.vmoptions" to configure custom JVM parameters.
  - A change of parameters in this file requires a restart of the application to take effect.
- "cpm_launcher.bat": File – Celonis Frontend launcher script
- "cpm_svc.exe": File – Celonis Frontend service wrapper
- "cpm_svc.xml": File - Celonis Frontend service wrapper configuration file
- "licenses.txt": File - contains licensing information

- “uninstall.exe”: File – Celonis uninstallation file.
- “vcredist_2008_x86.exe”: File – Microsoft Visual Studio 2008 Redistributable 32-bit library, included with the installer.
- “vcredist_2010_x64.exe”: File – Microsoft Visual Studio 2010 Redistributable 64-bit library, included with the installer.
- “vcredist_2015_x64.exe”: File – Microsoft Visual Studio 2015 Redistributable 64-bit library, included with the installer.
- “compute”: Directory – Contains all files necessary to manage the Compute processes
  - “logs”: Directory – Contains the Celonis Compute log files.
  - “root”: Directory – Contains all the Celonis generated Compute application files. From the Operations Guide perspective, the following files are relevant:
  - “temp”: Directory – Contains temporary Compute application files.
  - “.compute.sys.vmoptions”: File – Contains system configuration of the JVM.
    DO NOT EDIT this file, as it will get overwritten in case of a product upgrade. Use the “compute.user.vmoptions” file instead.
  - “application.properties”: File – Celonis configuration file that contains all the installation parameters.
    DO NOT EDIT this file, as it will get overwritten in case of a product upgrade. Use the “application-custom.properties” file instead.
  - “application-custom.properties.sample”: File – Sample Celonis configuration file that can be used as a starting point to change the Compute configuration settings.
    - Copy and rename this file to “application-custom.properties” to configure custom values.
    - A change of parameters in this file requires a restart of the Compute Service to take effect.
  - “compute.jar”: File – Compute Service with dependencies.
  - “compute.user.vmoptions.sample”: File – Sample JVM configuration file that can be used as a starting point to insert custom JVM parameters.
    - Copy and rename this file to “compute.user.vmoptions” to configure custom JVM parameters.
    - A change of parameters in this file requires a restart of the application to take effect.
  - “compute_svc.exe”: File – Celonis Compute service wrapper
  - “cpm_launcher.bat”: File – Celonis Compute launcher script
  - “compute_svc.xml”: File – Celonis Compute service wrapper c
  - “logging.xml”: File – Spring log configuration file for the Compute Service.

LINUX SYSTEMS

- “component_configurations”: Directory – Contains specific component configuration files (in a new installation containing only sample files) that can be used to address specific component settings for access, audit, loader-pools, login and trace logging, password rules and query-definitions.
- “jre”: Directory – Contains the AdoptOpenJDK JRE embedded package.
- “lib”: Directory – Contains java classpath runtime libraries
- “logs”: Directory – Contains the Celonis application log files.
- “logging.xml” - File that contains the default logging configuration
- “pdf-exporter”: Directory – Contains the PDF Exporting capabilities functions.
- “run”: Directory – Contains the PID file.
- “cpm.jar”: File – Celonis service dependency file.
- “config.properties”: File – Celonis configuration file that contains all the installation parameters.
  - **DO NOT EDIT** this file, as it will get overwritten in case of a product upgrade. Use the “config-custom.properties” file instead.
- “config-custom.properties.sample”: File – Sample Celonis configuration file that can be used as a starting point to change the Celonis Application Server configuration settings.
  - Parameters related to the Celonis Application Server listening interface, port, SSL, logging, SAML or multi-server deployment can be defined here.
  - Copy and rename this file to “config-custom.properties” to configure custom values.
  - A change of parameters in this file requires a restart of the application to take effect.
- “start.sh”: File –Celonis startup script.
- “stop.sh”: File –Celonis stop script.
- “start_application.sh”: File - frontend application startup script.
- “stop_application.sh”: File - frontend application stop script.
- “licenses.txt”: File - contains licensing information.
- “cpm.user.vmoptions.sample”: File – Sample VM options configuration for the Celonis service.
- “.cpm.sys.vmoptions”: File - System read-only CPM VM options configuration
  - **DO NOT EDIT** this file as it can contain sensitive options which are required for the successful CPM service startup.
- “.last-version.run”: File – contains the last launched version of CPM.
- “root”: Directory – Contains all the Celonis generated application files. From the Operations Guide perspective, the following files are relevant:
  - “uploads” – containing all files uploaded into Celonis (images, transports, .XLS, .CSV)
  - “backup” – containing all the configuration store backup snapshots.
- “compute”: Directory – Contains all files necessary to manage the Compute processes

- "logs": Directory – Contains the Celonis Compute log files.
- "root": Directory – Contains all the Celonis generated Compute application files. From the "temp": Directory – Contains temporary Compute application files.
- "application.properties": File – Celonis configuration file that contains all the installation parameters.
  - **DO NOT EDIT** this file, as it will get overwritten in case of a product upgrade. Use the "application-custom.properties" file instead.
- "application-custom.properties.sample": File – Sample Celonis configuration file that can be used as a starting point to change the Compute configuration settings.
  - Copy and rename this file to "application-custom.properties" to configure custom values.
  - A change of parameters in this file requires a restart of the Compute Service to take effect.
- "compute.jar": File – Compute Service with dependencies.
- "logging.xml": File – Spring log configuration file.
- "start_compute.sh": File - Compute startup script.
- "stop_compute.sh": File - Compute stop script.
- ".compute.sys.vmoptions": File - System read-only VM options configuration
  - **DO NOT EDIT** this file, as it can contain sensitive options which are required for the successful Compute service startup.
- "compute.user.vmoptions.sample": File – Sample VM options configuration for the Compute service.

TECHNICAL CONFIGURATION – MULTI-SERVER

This section shows which steps are necessary to configure the Multi-Server Deployment for a scale-out architecture. Since Celonis Process Mining 4.7, major parts of the configuration are performed in the new Compute Management UI, located in the System Settings.

Note: A Celonis user account with the **System Administrator** role is required to access the System Settings.

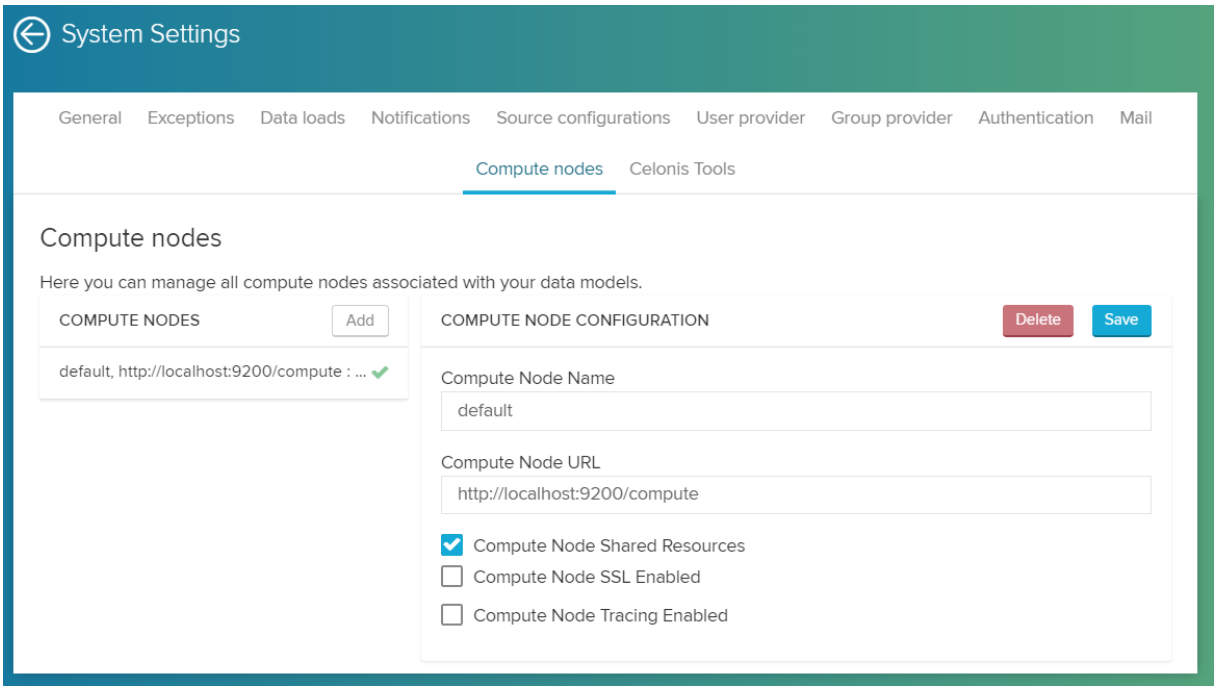

Figure 3: Compute Management UI

The following steps are necessary to configure the Multi-Server Deployment:

1. Install the Central Application Server and all Compute Services as described in the Installation Guide
2. It is recommended to change the default communication secret between these services to a custom secret. Also SSL can be configured. See the chapter SECURE COMMUNICATION BETWEEN CENTRAL APPLICATION AND COMPUTE SERVICE
3. **[Optional]** Change the ports of the Compute Services. To change the port for its default value, adopt the line server.port in the “application-custom.properties” file
4. **[Optional]** To define a custom directory that has write access to write the application data for the Compute Service, edit the line “fs.root” in the “application-custom.properties” file of the Compute Service
5. **[Optional]** To change the database of the configuration store, see the Configuration Store Guide
6. Create the Compute Services in the Compute Management UI (*System Settings* → *Compute nodes*) of the Central Application. This includes all Compute Services setup on separate servers. The default Compute on the Central Application Server should appear in the UI out of the box. For all following Compute Nodes you might want to configure, please perform the following setup in the Compute Management UI:
    a. Set a name for the Compute. The Compute names can be chosen freely, but should not contain special characters

b. Enter the Compute URLs defined in the respective “application-custom.properties” file. The URL format for the Compute Service is *http://<ip>:<above defined port>/compute*.
c. Configure the shared resource indicators. All Compute Services that run on the Central Application server shall have the option active. All Compute Services that run on separate servers shall have the option unticked
d. **[Optional]** Activate the tracing option for the Compute Service and specify the desired log level

Note: We **do not** recommend not to run multiple Compute Services on the same server.

7. Restart the Compute Services
8. Once configured, the Compute Servers can be selected for the execution of Data Model loads in the “Loading” menu of the Data Models

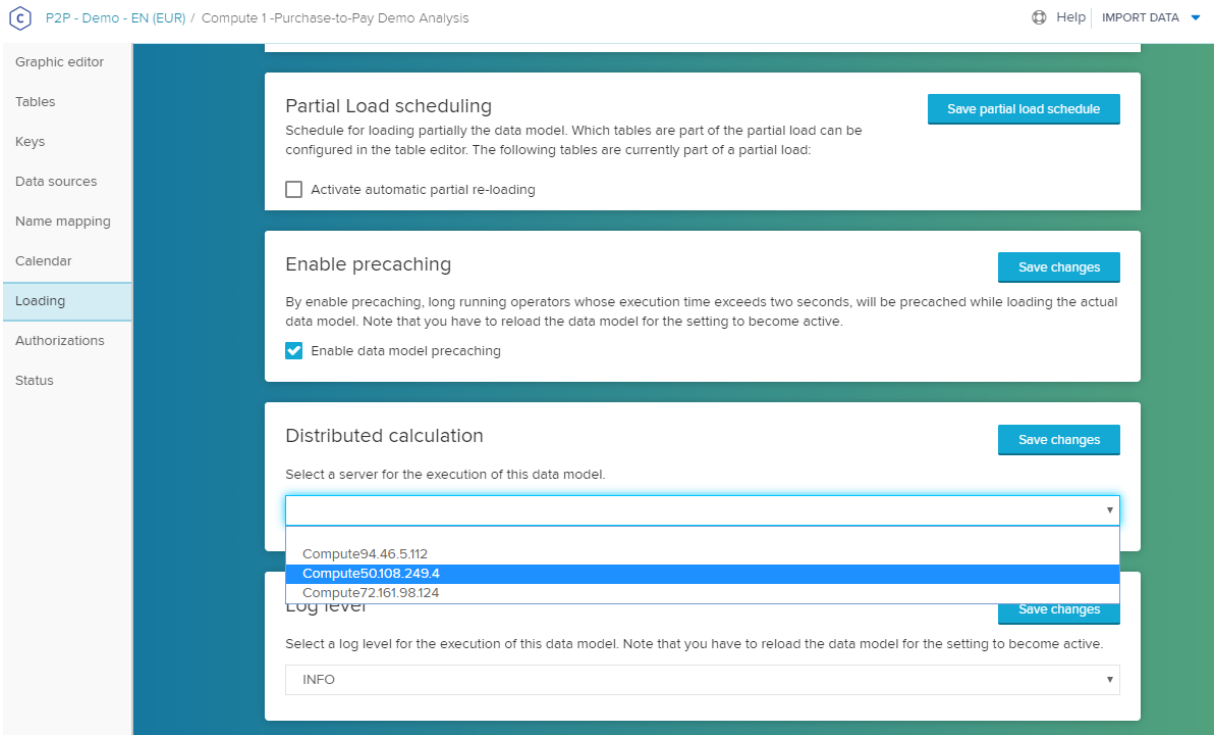

Figure 4: Compute Server configuration for each Data Model

TECHNICAL CONFIGURATION – SECURITY

GENERAL SECURITY

Celonis application provides built-in security for database connectivity. All user passwords in the Configuration Store are encoded (using SHA-256). Passwords for the connection to the analytics database are encrypted (using AES).

By default, the integrated Celonis Configuration Store powered by HSQLDB is secured with a password that is automatically generated. This password is not visible to the user and it cannot be read in any way. It is simply embedded within the application. If you want to override this setting, you can do so by editing the Celonis Configuration Store Settings from the "config-custom.properties" file in your installation directory (**this can only be configured before the first startup of the application**).

We recommend setting up the Celonis Configuration Store using a separate database system. For more information, please refer to the Celonis Configuration Store Setup Guide.

If you do not want to store the password for your custom Celonis Configuration Store in plain text in the "config-custom.properties" file, you can use the Celonis Key Vault. Therefore, please configure the path to the generated private key file which the server uses to open the vault. Make sure that the key file is only readable by the service user and not by anyone else. For more information on how to generate the private key and encrypted passwords, please refer to the Celonis Manual.

The Celonis web access security relies on the Spring Security Framework hardening. As Celonis can also make use of up to date security standards, it is recommended for you to enable and use the SSL option right from the beginning, after the installation. This feature can be enabled as well from the "config-custom.properties" file. Upon enabling the SSL feature, you must go through the following steps:

- Set the "server.ssl" option to "true"
- If there is no keystore available, create a Java keystore. To generate a key in a local keystore, please use the Java keytool[1] or import an existing key. A sample command for generating a new key is:
  - **`keytool -genkey -alias celonis4 -keyalg RSA -keystore keystore.jks -keysize 2048`**
  - Please note that for paths on windows, you should use forward slashes (e.g. `E:/celonis/my_keystore.jks`). Please refer to the Oracle Manual for more information
- Generate a new CSR and/or import the CRT (existing or obtained from the CA after signing the CSR) into the keystore. For more information, the same documentation from the previous step can be used
- Provide the keystore file path using the "server.ssl.keystore" parameter

[1] Please make sure to use the keytool utility provided with the Celonis installation in "<installDir>/jre/bin/keytool"

- Specify the keystore alias using the “server.ssl.keyalias” parameter. The key alias name was provided upon the keystore creation
- Specify the keystore password using the “server.ssl.keystorepw”. This password is required to open the keystore
- Specify the private key password using the “server.ssl.keypw”. This password is required to read the private key
- *[Optional]* In case the allowed SSL protocols and/or ciphers need to be limited, please specify the following options:
    - “server.ssl.enabledProtocols”: comma separated list of *allowed* SSL protocols, e.g. `TLSv1.2,TLSv1.3`
    - “server.ssl.ciphers”: comma separated list of *allowed* SSL ciphers, e.g. `TLS_CHACHA20_POLY1305_SHA256,TLS_AES_128_GCM_SHA256`
    - If any of these properties are unspecified (commented) or empty - this implies **all** (e.i. all default protocols and/or ciphers, supported by bundled web server, will be allowed)
    - To get a list of most commonly used SSL protocols and ciphers one could use command `openssl ciphers -s -stdname`

During the installation process, the password for the default user “sysadmin” is requested. Please make sure that you are going to use a secure password. If there is no password specified, the installer will choose the default “$admin!” password. We do not recommend keeping the initial password for the “sysadmin” in a productive environment, thus this password should be changed as soon as possible via the web frontend. The default password policies also force you to change the password directly after the first login. The password policies are also highly customizable from the “password-rules.properties” file. There you can enable or disable the rules and set password minimum requirements such as minimum length, complexity and change rate. With the number “0” the options can be set to “unlimited”, for example “password.rules.last_passwords_forbidden=0” means that any old password may be reused.

Authorization in Celonis is done via the Authorization Objects. They can be used to automatically filter the dataset for users and groups. This can be particularized for each user and dataset. With this functionality, the administrator can opt for only showing certain parts of the data to be displayed to certain users or groups in such a way, that the users or groups will not notice that they are having access to incomplete data. This grants the perfect layer of data protection and privacy for customer’s data. The authorized SQL queries can be defined in the Celonis GUI. If you want to use an external user permission system, it can be helpful to disable all permission sharing functionality for all regular users. Only content administrators are then allowed to pass permissions to users and groups, when this setting is enabled. To enable this setting, please set the option “instance.disable_user_permissions” in the “config-custom.properties” to “true”.

For a secure network setup, we recommend using a dedicated server and close all ports, but the ones required by our application. In the case in which another web server will run in front of the Celonis Application Server, the server port can be bound, for example, to the localhost. This can be achieved from the “config-custom.properties” file using the “server.interface” and “server.port” parameters. Even more, all connections with the database can be encrypted. This can be done using the JDBC string by adding the “encrypt=true” parameter. In case your analytics database installation uses a self-signed certificate, you need to add the “validateSSLcertificate=false” parameter. For more information, please consult the official documentation of your Analytics Database.

SECURE COMMUNICATION BETWEEN CENTRAL APPLICATION AND COMPUTE SERVICE

Custom communication secret

The communication between the Central Application and a Compute Service is secured by a security token. During the standard installation and update, a communication secret is generated. It is recommended to change this default secret to a user defined secret:

1. Stop the Central Application Service and the Compute Services
2. Create the “config-custom.properties” file by copying the “config-custom.properties.sample” file, if not yet existent
3. Uncomment the line jwt.secret and add your custom secret
4. For each Compute Service, create the “application-custom.properties” file by copying the “application-custom.properties.sample” file
5. Uncomment the line *security.jwt.secret* and enter the above defined custom secret
6. Start the Central Application Service and the respective Compute Services

Secure Sockets Layer (SSL)

If the communication between the Central Application Service and one or multiple Compute Services is not trusted, the connection can be secured via SSL.

To set up an SSL connection between the Central Application Service and a Compute Service, follow the steps below. In the case of a Multi-Server-Deployment, SSL needs to be configured for every respective Compute Service.

1. Stop the respective Compute Service
2. If already created, open the “application-custom.properties” file in the installation directory of the Compute Service

3. Uncomment the following lines and enter the necessary information:
    - Set the option “server.ssl.enabled” to “true”
    - Provide the keystore file path using the “server.ssl.key-store” parameter. If there is no keystore available, create a Java keystore. To generate a key in a local keystore, please use the Java keytool[2] or import an existing key (see chapter GENERAL SECURITY)
    - Uncomment the parameter “server.ssl.key-store-type”
    - Specify the keystore password using the “server.ssl.key-store-password”. This password is required to open the keystore
    - Specify the keystore alias using the “server.ssl.key-alias” parameter. The key alias name was provided upon the keystore creation
4. Save and close the file
5. Open the Compute Management UI in the Celonis frontend
6. Set the option “Compute Node SSL Enabled”. In the case of a Multi-Server Deployment, this option has to be enabled for every Compute Service separately.
    - Specify the truststore URL for every Compute Service
    - Specify the truststore password for every Compute Service
7. Save the changes and restart every Compute Service

SECURING PROPERTIES IN CONFIGURATION FILES

By default, the property values in Celonis 4 configuration files (e.g., *config-custom.properties*) are entered in plain text. However, multiple options to secure those values can be used:

```
my.property=plain:mypassword
my.property=base64:bXlwYXNzd29yZA==
my.property=env:MY_VARIABLE
my.property=env64:MY_VARIABLE
```

Figure 5: Password options

Base64 encoding

Encoded properties can be entered by using the **`base64:`** prefix, followed by the base64 encoded property value. Please note that base64 encoding is not an encryption technique.

[2] Please make sure to use the keytool utility provided with the Celonis installation in “<installDir>/jre/bin/keytool”

Environment variables

Set up a system environment variable according to your operating system. Reference the variable in the property value with the **env:** prefix.

Environmental base64 encoding

This method combines base64 encoding and the use of environment variables:

1. Encode the password with base64
2. Set up an environment variable with value of the base64 encoded password
3. Reference the variable name in the configuration file with the prefix **env64:**

As an alternative to the security measures above, the property values can also be retrieved from CyberArk. For detailed information, please refer to the chapter TECHNICAL CONFIGURATION – CYBERARK AAM INTEGRATION.

PYTHON SECURITY

To use PI Machine Learning, you can use the Python API of Celonis. The access to the API is restricted to authenticated users. To eliminate the need to store the user's password in the calling python script, API keys can be generated within the application to allow programmatic access to the API. To generate an API key, the user must access his profile, where he can see all current API keys, create new API keys and delete existing API keys for his user. API keys should not be shared between users and it is recommended to create a separate technical user for using the Python API.

In order to completely disable API authentication for all users, set the property *apiKey.authentication.active* in the "config-custom.properties" to *false* and restart the application. All existing API keys will be disabled temporarily and no new keys can be created.

TECHNICAL CONFIGURATION – HIGH AVAILABILITY (HA)

This section shows which steps are necessary for Celonis to operate in a High Availability environment.

The Celonis application can be installed in a High Availability Cluster configuration to benefit from:

- High Celonis Application Server uptime
- Resource scalability
- Migration easiness

It is recommended to use a dedicated VM server for Celonis and to perform regular snapshots to this VM on a remote location.

As Celonis works highly intensively with the analytics database, its performance and ability to often send requests to the Database Server(s) highly depends on the Database Server(s) performance and availability. As such, it is recommended that the Database Server(s) should operate within a High Availability – High Performance clustering environment and that the fastest communication wiring and protocols with the Celonis Application Server are assured. Ideally, the analytics database environment can make use of clustering configurations. It is recommended to scale the Database Server(s) accordingly with the database size and complexity.

Due to the large number of infrastructure concepts only a sketch is displayed in the figure below. This is not to be considered as an infrastructure design, but it should give you an overview of the key components you should consider while using Celonis in a HA design. Networking elements and connectivity are also completely excluded from this diagram. For more information, the specific solution and/or vendor's HA design must be consulted.

The Database Servers and High Availability Cluster's security needs to be applied according to specific tools provided by the analytics database software and/or by the High Availability Cluster's vendors and their support, considering each IT Infrastructure specific security policies.

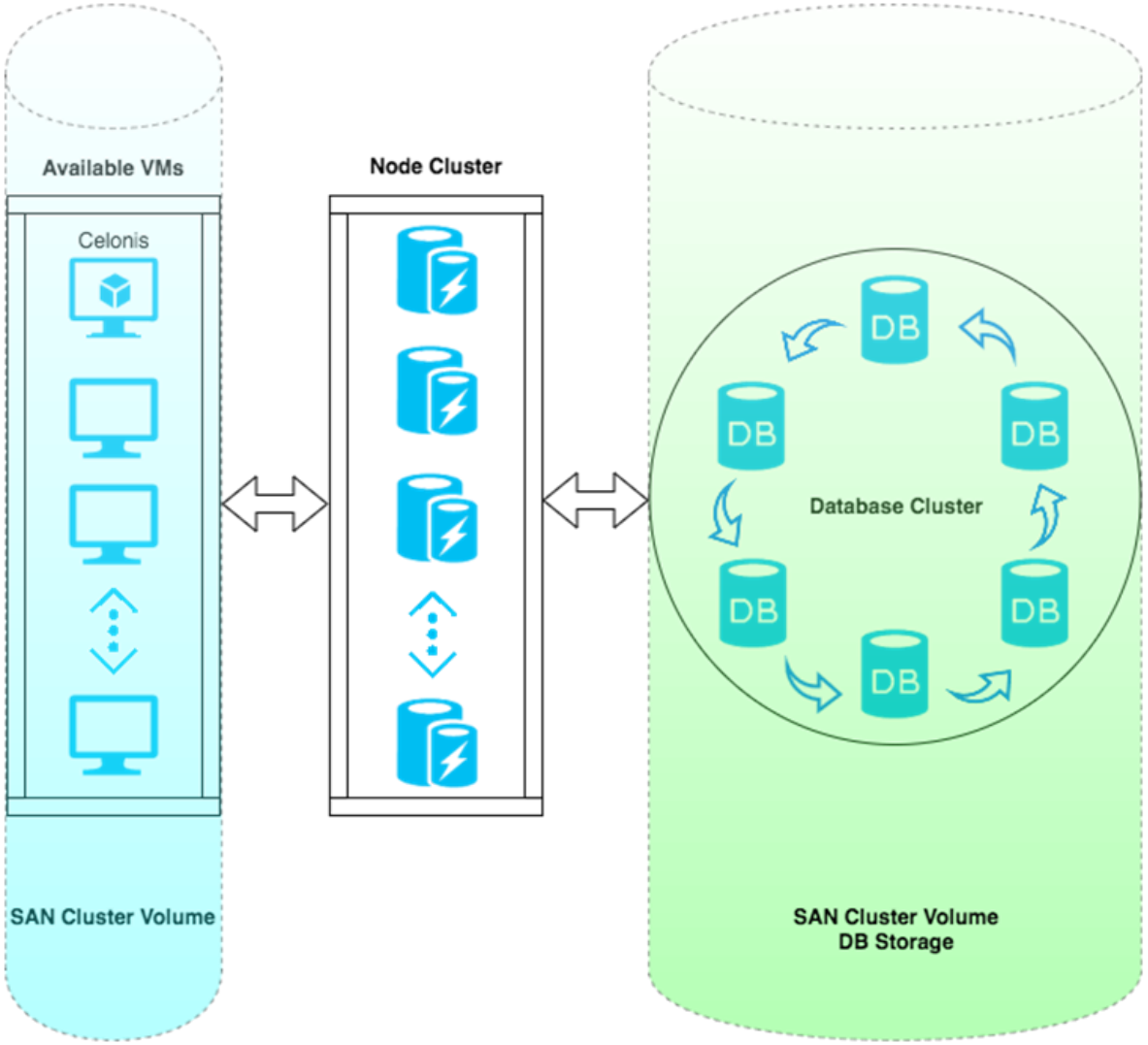
Available VMs
Celonis
Node Cluster
DB
DB
DB
Database Cluster
DB
DB
DB
SAN Cluster Volume
SAN Cluster Volume
DB Storage

Figure 6: HA-1

TECHNICAL CONFIGURATION – LOGGING

LOGGING FOR CELONIS

For specific interactions in Celonis - like *logging in* or *creating a user* - Celonis allows the creation of logs.

Data Model Logs

For Data Model interactions, there are pre-configured levels of granularity at which logs are created. The log level can be configured in the frontend view of the Data Model under the “Loading” tab. The existing log levels are “INFO”, “DEBUG”, “WARN” and “ERR”:

- INFO - basic information. *This option is recommended to limit the log storage size*
- DEBUG
- WARN
- ERR

Audit Logging

Deprecation Note: The former audit logger that is configured using the “audit-logging.properties” file is deprecated. Therefore, the advanced audit logger described below has to be used since version 4.7.2.

By default, no audit logs are written, but the events can be activated individually. To enable the configuration, copy the “component_configurations/audit-logging-advanced.properties.sample” file in your installation path and rename it to “audit-logging-advanced.properties”. Then the desired events can be enabled by setting the options from “false” to “true”.

Note: The top-level property for each object+action type is named “*.*.object_audit_enabled”. It acts as a primary switch and needs to be set to “true” for the related audit object+actions to be enabled.

Examples:

Scenario 1: *Authentication* action type is **disabled**, log for *login event* is **disabled.** Nothing will be logged.

- audit_logging_advanced.authentication.object_audit_enabled=**false**
- audit_logging_advanced.authentication.login=false

Scenario 2: *Authentication* action type is **disabled**. The *login event* switch is **ignored** and nothing will be logged.

- audit_logging_advanced.authentication.object_audit_enabled=**false**
- audit_logging_advanced.authentication.login=true

Scenario 3: *Authentication* action type is **enabled**, but the *login event* action is still **disabled**. Only *failed login* events are logged.

- audit_logging_advanced.authentication.object_audit_enabled=**true**
- audit_logging_advanced.authentication.login=**false**
- audit_logging_advanced.authentication.failed_login=**true**

Scenario 4: *Authentication* action type is **enabled**, both *login* and *failed login* events are logged

- audit_logging_advanced.authentication.object_audit_enabled=**true**
- audit_logging_advanced.authentication.login=**true**
- audit_logging_advanced.authentication.failed_login=**true**

Formatting:

The advanced audit logs have the following format, separated by semicolons:

- Date-time (`yyyy-MM-dd'T'HH:mm:ss.SSSXXX`)
- Host Information (`audit_logging_advanced.host_info.object_audit_enabled=true`) [Optional]
- Operator (user performing the event)
- Object Type (e.g. User, Data Model, …)
- Object Property Type (nested objects like Data Model → Loading) [Optional]
- Action Type (e.g. Create/Remove/Update/Delete)
- Additional Action Type (nested actions corresponding to the Object Property Type) [Optional]
- Object Value
- Description

Properties marked with [Optional] only apply to a subset of logged events. Another important point is the difference between Object Type and Object Property Type as well as Action Type and Additional Action Type. While the combination of Object Type and Action Type describes the overarching action of the user, the Object Property Type and the Additional Action Type add more detail in case of nested actions. For example, a user is creating a new database connection for a data model.

- Object Type = Data Model

- Action Type = Update
- Object Property Type = Data source
- Additional Action Type = Create
- Object Value = Created data source connection details

Individual options can be enabled or disabled for each of the following Object Types:

- User
- Group
- Authentication
- API Key Authentication
- API Key
- Authorization
- User Authorization
- Data Model Authorization
- All Objects (Analysis, Data Model, Project, Folder)
- Data Model
- Analysis
- Transport
- System Settings
- Host Information (Hostname, IP, Port)

The Action Type that is logged can be generic or specific to the Object Type:

- Generic Actions
    - Create
    - Update
    - Rename
    - Delete
    - Move
    - Open
- Authentication
    - Login
    - Failed Login
    - Logout
- API Key Authentication
    - Login
    - Failed Login
- API Key
    - Create
    - Delete
- Authorization
    - Permissions updated
    - Permissions denied

    - Assign
    - Revoke
    - Assign Group
    - Unassign Group
- User
    - Lock
    - Unlock
- Analysis
    - Publish
    - Update Data Model
- Transport
    - Import
    - Export
    - Download
- Data Model
    - Edit
    - Load
    - Unload
    - Cancel Load
- System Settings
    - Update Notifications
    - Update Source Configurations
    - Update Providers
    - Update Authentications
    - Update Mails
    - Update Compute Nodes

With release 4.7.2, we support audit logging to external databases to enable advanced analysis of the logging events.

You can find the required new properties in the audit-logging-advanced.properties.sample file in <installDir>/component_configurations after the update. If you are already using the advanced audit logs, you can copy the new properties into the existing audit-logging-advanced.properties file:

```
# Logging to the database is disabled by default
# To enable logging to the database, uncomment the next section, set audit_logging_advanced_config.database to true
# and configure a datasource
# audit_logging_advanced_config.database=true
# audit_logging_advanced_config.database.driverClassName=<driver>
# audit_logging_advanced_config.database.url=<url>
# audit_logging_advanced_config.database.username=<user>
# audit_logging_advanced_config.database.password=<password>
```

Login Logging

Logging at what time a user has logged into Celonis software is also possible. By default, this feature is turned off, but it can be enabled by copying the “login-logging.properties.sample” file to “login-logging.properties” and fill out the required information:

- “Login_logging.enabled”: Either false or true
- “Login_logging.database.url”: As the information is saved within a database, the JDBC connection URL must be entered here
- “Login_logging.database.driver”: The JDBC driver used to connect
- “Login_logging.database.user”: The database user with proper access rights
- “Login_logging.database.password”: The database user’s password
- “Login_logging.database.success_query”: The query that will be executed in case of a successful login

Please note that in case you activate the login and/or audit log, personal information on the users of the application (username, user ID, first name, last name, email address) will be stored in the respective log files and/or database tables. You as a customer are responsible to adhere to the Data Protection Principles for this collected data, e.g. related to deletion of personal data. To delete login logs, use the standard database functionality of deleting rows in tables. To delete audit logs, use the standard file system mechanisms of deleting text from files or deleting whole files.

TECHNICAL CONFIGURATION – WRITABLE ROOT

With version 4.7.1, we slightly adapted the configuration of the filesystem.writableroot property:

The filesystem.writableroot in *config.properties* is set to the default installation directory in the installer (for Linux *<installDir>/root* and for Windows *<installDir>/appfiles)*, and it always overrides this property on upgrade. If the target or the subdirectory that points to the writableroot are manually changed, two writableroots are created during the update and the configuration can be lost. To prevent such a scenario, we introduced a new property called *filesystem.writableroot.allow_different_dir* in the configuration files.

With this new property, the writable root configuration behaves as follows:

Application Startup

- If *filesystem.writableroot* does not point to the default directory and *filesystem.writableroot.allow_different_dir=false*, the application will not startup with a clear message highlighting the root cause.
- If *filesystem.writableroot* does not point to the default directory and *filesystem.writableroot.allow_different_dir=true*, the application startup will succeed with a warning message printed out in the logs.

Application Update

- If *filesystem.writableroot* does not point to the default directory and *filesystem.writableroot.allow_different_dir=false*, the application update will overwrite writableroot to point to the default directory. The customization has to be restored after the update.
- If *filesystem.writableroot* does not point to the default directory and *filesystem.writableroot.allow_different_dir=true*, the application update will preserve the custom writableroot.

Important Notes:

- Changing the default filesystem.writableroot is not recommended. There are very few cases where it makes sense.
- If the property should be customized, please do so in config.properties. References in config-custom.properties should be removed.

TECHNICAL CONFIGURATION – CYBERARK AAM INTEGRATION

With the integration of the CyberArk Application Access Manager (AAM), the Celonis end-user does not have to specify sensitive database credentials into the configuration files or the application anymore.

Prerequisites

- The *javapasswordsdk.jar* runtime library supplied by CyberArk has to be placed in the *<installDir>/lib* folder within the Celonis 4 installation directory.

- 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. For this part, please request the respective documentation from CyberArk directly.

Placing the javapasswordsdk.jar in the correct directory

1. Stop the Celonis Process Mining application
2. Retrieve the *javapasswordsdk.jar* from your CyberArk installation, typically:
    a. */opt/CARKaim/sdk/javapasswordsdk.jar* for Linux,
    b. *C:\Program Files (x86)\CyberArk\ApplicationPasswordSdk\JavaPasswordSDK.jar* for Windows
3. Place the *javapasswordsdk.jar* runtime library in the *<installDir>/lib* folder in the Celonis 4 installation directory
4. Restart the Celonis Process Mining application

Password retrieval - configuration files

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.

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
```

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 of the exact version you provided in *<installDir>/lib* 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`

Password retrieval - frontend

The frontend configuration follows the same rules & notesas the configuration of the properties. The following frontend password can be retrieved from CyberArk:

Database connections:
The "password" to connect to a database from within a Data Model.

Source configurations:

- “LDAP password” in System Settings ⟶ Source Configurations ⟶ LDAP Sources
- “Database password” in System Settings ⟶ Source Configurations ⟶ Database Sources

SMTP Server configuration
SMTP Server Password in System Settings ⟶ Mail

APPLICATION SERVER ADMINISTRATION

Since the application will be running as an operating system service, this part describes how to correctly configure it as such. It will also describe the necessary tools for administration.

REQUIRED TOOLS

The following tools are needed on the Celonis Application Server to successfully administer the Celonis application:

- A text editor

All supported operating systems provide these tools out of the box. Furthermore, the standard Linux command line tools (like tail, grep and others) will help you in accessing log and configuration files.

As Windows lacks most of those command line tools and the built-in text editor is lacking features like syntax highlighting or support for UNIX-style line breaks, it is recommended to install specific tools for Windows (e.g. Notepad++, baretail, baregrep).

For administrative tasks inside the software itself a web browser is required. As the application can normally be accessed from outside the server, there is no direct need to have a web browser on the Celonis Application Server itself. It could however be beneficial to test connection issues, etc.

CONFIGURATION FILES

Central Application

The basic Celonis server configuration takes place during the installation process. The central configuration file of the Celonis Central Application is “config.properties”. This file can be found inside the root directory of the installed software. **You should never manually edit this file**. The file gets overwritten in the update process. All user custom configuration should be made in the “config-custom.properties” file. Further information can be found in the sample configuration file “config-custom.properties.sample”.

Compute Services

All Compute Service configurations are stored in the “application.properties” file in the “<installDir>/compute” directory. **You should never manually edit this file**. To edit the configuration of the Compute Service, you can create a “application-custom.properties” file. For the Compute Service there is also a sample file with further information in “application-custom.properties.sample”.

Component-specific configuration

Component specific configuration files as well as the respective sample files can be found in "<installDir>/component_configurations".

CUSTOM JAVA OPTIONS

Central Application

Additional java options can be applied, for example to increase the java heap memory or to configure the JAVA MANAGEMENT EXTENSIONS as described here.

To add additional java options, add them to the "cpm.user.vmoptions" file. If it is not existent, please create it by creating a copy of the "cpm.user.vmoptions.sample" file located in the installation directory. Rename the copy to "cpm.user.vmoptions" and apply the required java properties. Application Server restart is required afterwards.

Compute Services

The same is applicable for the Compute Service. To add additional java options, add them to the "<installDir>/compute/compute.user.vmoptions" file. If it is not existent, please create it by creating a copy of the "compute/compute.user.vmoptions.sample" file located in the installation directory. Rename the copy to "compute.user.vmoptions" and apply the required java properties. Compute Server restart is required afterwards.

DEPLOYING ADDITIONAL JDBC DRIVERS

General deployment procedure

The Celonis Process Mining application is using JDBC to connect to databases, e.g., the Analytics Database. Depending on the database type, Celonis does not provide the JDBC driver, and it has to be deployed by the customer. We indicate those database types with the note "additional driver required":

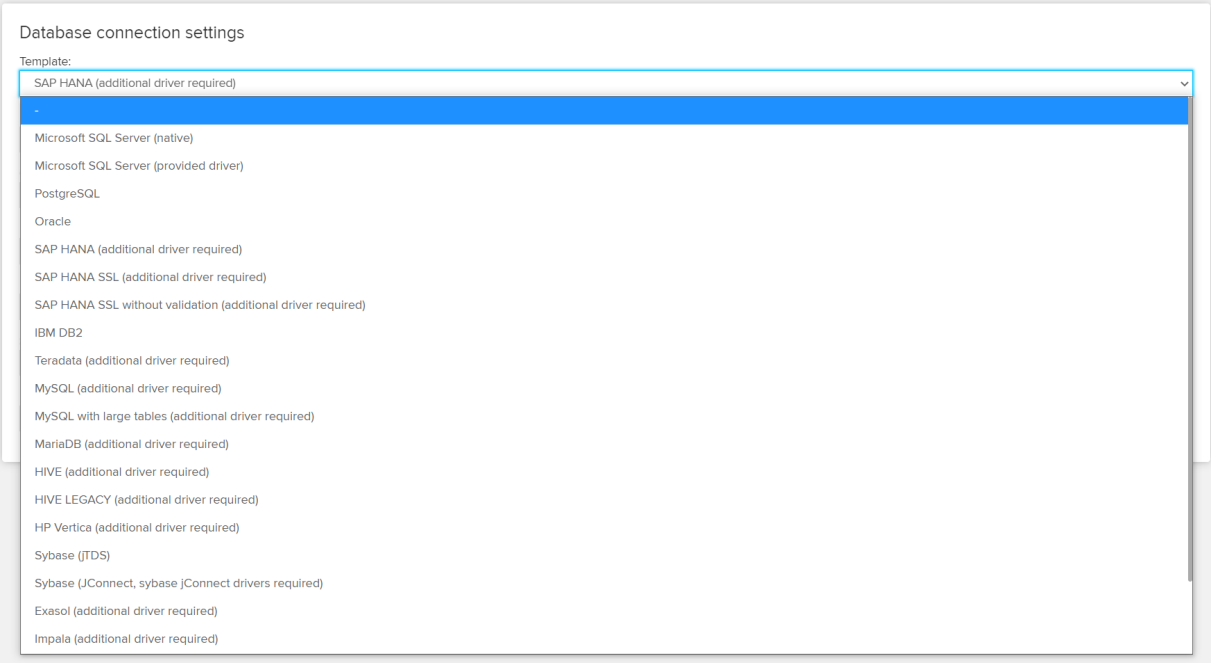

Figure 7: Example - Database Connections

Please follow the steps below to deploy a custom JDBC driver:

1. Acquire the JDBC driver .jar file for the respective database from your database vendor. **Please make sure to use a driver version that is compatible with the database version.**
2. Stop the Celonis application
3. Move the JDBC driver into the *<installDir>/lib* folder in the Celonis installation directory.
   If custom libraries location is required - customer could override property `loader.path` in "cpm.user.vmoptions", like:
   `-Dloader.path=<absolute_path_to_installDir>/lib`
4. Restart the Celonis application and try to connect to the respective database.

Note: Due to recurrent incompatibility between driver versions and customer databases, the SAP HANA JDBC driver (since 4.7.1) and the Amazon Athena JDBC driver (since 4.7.2) are not distributed with the Celonis 4 application anymore.

Microsoft JDBC Driver for SQL Server Authentication Library for Windows

We removed the **`mssql-jdbc_auth-<version>.x<64/86>.dll`** (Microsoft JDBC Driver for SQL Server Authentication Library for Windows) from the embedded dependencies with version 4.7.3. In case the application is running on Windows and the SQL Server Authentication (indicated by

integratedSecurity=true in the JDBC string) is used to connect to the Celonis Configuration Store or the Analytics Database, the library has to be deployed manually:

1. Stop the application
2. Download the respective driver version based on your SQL Server version and compatibility from here. Please follow the Microsoft recommendations in case of any questions: here.
3. Place the library in **<installDir>\jre\bin**
4. Start the application

CELONIS AS AN OPERATING SYSTEM SERVICE

Using the Jetty Embedded Application Server, the Celonis application is installed as a service inside the Windows Operating System, offering the possibility to be administered as any other regular OS service.

The Celonis Service name can be customized in the respective configuration files. The usual service name that is used by Celonis during the installation process is “Celonis CPM 4 frontend” for the Central Application Service and “Celonis CPM4 compute” for the Compute Service. For Windows operating systems, the Celonis Services can be configured using the following startup types:

- “Automatic (Delayed Start)” (Recommended),
- “Automatic”,
- “Manual” or
- “Disabled”

For Linux operating systems, the Celonis Application Services can be manually controlled using the “start.sh” and “stop.sh” bash scripts. In case of a Multi-Server Deployment, the Compute Services on separate Compute Servers can be controlled using the “start_compute.sh” and “stop_compute.sh” bash scripts. Using OS specific methods, these scripts can be set to automatically run in special conditions (e.g. automatically start the software on computer boot). Example scripts are provided in “<installDir>/scripts”.

The Celonis Services can receive the following service commands:

- On Windows: “Start”, “Stop” or “Restart”
- On Linux: controlled via the provided bash scripts

A service restart, if needed, could also be performed by first stopping and then starting up the service. To offer flexibility, Celonis does not require the operating systems service installation to run. The Celonis Application Server can also be run manually, only when it’s needed, however this is not recommended in productive environments. We highly recommend using Celonis installed as an operating system service to benefit from ease in administration. Operating systems services are also offering the possibility that when no longer needed, they can be uninstalled.

PERIODICAL TASKS – ARCHIVING FILES

This section describes how to best manage the task of periodically archiving old files: The log and backup files of the application server as well as new releases of Celonis.

Using Celonis for a long period of time will put you in the situation of dealing with old files. Old files will take unnecessary disk space and keeping old files mixed with current files will make the administration process more and more difficult overtime. We recommend archiving these old files and/or even set up an “Old Files Strategy” policy.

The archiving policy should consider the following cases:

CELONIS LOG FILES

Central Application logs (Celonis CPM4 Frontend)

The log files generated by the Celonis Central Application Service are in the “logs” folder that resides in the root of the Celonis Server install path. The “logs” folder will contain the following log files types:

Windows

- **`<installDir>/logs/cpm-stdout.log`**
  Celonis CPM4 Frontend startup messages (before full application server start)
- **`<installation_directory>/logs/cpm_svc.err.log`**
  `<installation_directory>/logs/cpm_svc.out.log`
  `<installation_directory>/logs/cpm_svc.wrapper.log`
  Windows Service Wrapper logs (when starting *Celonis CPM4 Frontend* service)
- **`<installation_directory>/logs/cpm.log`**
  Celonis CPM4 Frontend general service log file for current date (today’s logs)
- **`<installation_directory>/logs/cpm.<date>.log.gz`**
  Celonis CPM4 Frontend archive service log files, rotated daily by default

Linux

- **`<installDir>/logs/cpm-stdout.log`**
  Celonis CPM4 Frontend startup messages (before full application server loading)
- **`<installDir>/logs/cpm.log`**
  Celonis CPM4 Frontend general service log file for current date (today’s logs)
- **`<installation_directory>/logs/cpm.<date>.log.gz`**
  Celonis CPM4 Frontend archive service log files rotated, daily by default

Compute logs (Celonis CPM4 Compute)

In addition to the logs files of the Central Application, every Compute Service will generate the following log files:

Windows

- **`<installDir>/compute/logs/compute-stdout.log`**
  Celonis CPM4 compute startup messages (before full compute server loading)
- **`<installation_directory>/compute/logs/cpm_svc.err.log`**
  `<installation_directory>/compute/logs/cpm_svc.out.log`
  `<installation_directory>/compute/logs/cpm_svc.wrapper.log`
  Windows Service Wrapper logs (when starting *Celonis CPM4 compute* service)
- **`<installation_directory>/compute/logs/compute.log`**
  Celonis CPM4 compute general service log file for current date (today's logs)
- **`<installation_directory>/compute/logs/compute.<date>.log.gz`**
  Celonis CPM4 compute archive service log files, rotated daily by default

Linux

- **`<installDir>/compute/logs/compute-stdout.log`**
  Celonis CPM4 compute startup messages (before full compute server loading)
- **`<installation_directory>/compute/logs/compute.log`**
  Celonis CPM4 compute general service log file for current date (today's logs)
- **`<installation_directory>/compute/logs/compute.<date>.log.gz`**
  Celonis CPM4 compute archive service log files rotated, daily by default

Rotate log files

As mentioned above, the main service logs for the Celonis 4 Frontend (**`cpm.<date>.log`**) and the Celonis 4 Compute (**`compute.<date>.log`**) are now rotated on a daily basis by default.

Therefore, for Linux users, we commend to remove the previous CPM 4 logrotate configuration from **`/etc/logrotate.d/`**

Custom logging configuration

The custom logging configuration was updated with version 4.7.3. It was previously configured in `<installDir>/config-custom.properties` via the property **`logging.config=`**. **This property is no longer supported**.

To apply a custom configuration you can either:

- Refer a custom logging config file using the JVM property **`-Dlogging.config`** in `<installDir>/cpm.user.vmoptions`
- Refer a custom logging configuration in the file **`application.properties#logging.config`**

residing in the same directory as the application jar.

Note: Starting from CPM4.7.3 the logging library was changed from *log4j2* to *logback*. Therefore, an update to the `logging.xml` configuration format could be required. For more information, please refer to: https://logback.qos.ch/manual/configuration.html

Retrieving engine log files

The engine log files can be found in "*<installDir>/compute/root/logs/celonis-service*". They have the following naming convention: "*<dataModelId>-<date>-<timestamp>.log*"

CELONIS RELEASES

Celonis gets periodic new builds that may deal with new features, tuning, customization, new web browser compatibility, bug fixes or up to date security standards. As such, we always recommend upgrading Celonis to the latest version. As the upgrading procedure describes, old Celonis production releases (the install kits) should not be deleted right away. They must be kept as backup versions in case the customer experiences problems with the newest release (the use of older web browsers for example). At some point in time, these old versions of Celonis will take unnecessary disk space. As we do not encourage you to delete anything unless you need to (you may not know when you will need something from the old files), we will make the following recommendations for an archiving strategy:

- Archive (.zip, .tar.gz, etc.) old log files once per month and thus keeping in the "logs" folder only log files newer than 30 days
- Move all old Celonis software installer releases inside an "Old" folder and thus keeping only the last two releases in your current Celonis installation path (outside the "Old" folder)
- Please note, to consult the upgrading procedure to be aware of all the files that are modified during upgrading to a new release – they should all be part of the old Celonis version archiving procedure. Usually we are taking care of this automatically, but there may be special releases at some point in time that will require some extra steps
- Move all old archives to a remote location to free up unnecessary used disk space on the current server

All Celonis recommendations should be treated as such and you should always consider first the digital files management policies already established by your company, if they are available.

CELONIS CONFIGURATION STORE BACKUPS

In case you are using the integrated Celonis Configuration Store powered by HSQLDB with its predefined backup policy, the backup directory might grow significantly over time. Please make sure to adjust your database backup retention policy in the “config-custom.properties” and/or take care of manually cleaning the outdated backup files.

BACKUP AND RECOVERY – BACKUP CELONIS CONFIGURATION STORE

Here, you will learn about how to regularly backup the Celonis Configuration Store as well as restore it in case of a failure.

The Celonis Configuration Store is the central storage of the application metadata, e.g. system settings, users, groups, definitions of analyses and Data Models and contains all configuration done via the web frontend. The actual data to be analyzed resides in the analytics database.

For small test and development installations, Celonis can make use of the integrated Celonis configuration store powered by HSQLDB. For this data store, there is an out of the box predefined backup policy. The Celonis configuration store is then automatically backed-up each night to the “appfiles/backup” folder in the root of the Celonis Server install path. We highly recommended keeping a remote backup of this folder. This will allow the possibility to restore the application metadata in case a disaster occurs. The backup is set to be performed online so you do not have to worry about any Celonis downtime during this procedure. The backup is run automatically every night at exactly 3 AM (while the application is running) and additionally whenever the Celonis service is started. All backups taken are full backups for all application metadata.

For medium to large installations as well as any productive installations, we recommend to set up the configuration store separately. Please note, that an automated backup of the Celonis configuration store is then not part of the application and must be implemented separately by the customer.

RECOVERING FROM A BACKUP FOR INTEGRATED CELONIS CONFIGURATION STORE

The backup files follow the naming convention “appdata-<yyMMdd>T<HHmmss>.tar.gz” with the timestamp indicating when the backup was started. Technically, this file is a zipped version of the full Configuration Store. To restore the backup, please do the following steps:

- Identify the backup you want to restore. Use the timestamp of the backup to identify the backup you want to restore
- Extract the “appdata-<yyMMdd>T<HHmmss>.tar.gz” file. It should contain four files called “appdata.lobs”, “appdata.log”, “appdata.properties”, “appdata.script”
- Shut down the Celonis Application Server. Go to the “services.msc”, identify the service (Default: “Celonis”) and stop it
- Identify the path where the active Configuration Store is located. Open the “config.properties” configuration file (or “config-custom.properties” if you have customized the Configuration Store location). Identify the property “filesystem.writableroot”.
- The path points to where the database files are located. These files should have the same names as the files contained in the extracted archive
- Create another backup of the current database files by simply copying them somewhere else.
- Copy the previously extracted files and overwrite the originals
- Start the Celonis Application Server service using the “services.msc“ console.
- Wait for the Celonis Application Server to be started completely

As a result, your backup is restored.

Please note, if a file called “appdata.lck” is present, it means that the service was not fully stopped and that the Configuration Store is still being used. Please make sure that the service is completely stopped before you restore the database files from a backup.

BACKUP AND RECOVERY – BACKUP ANALYTICS DATABASE

Here, you will learn about how to regularly backup the analytics database as well as restore it in case of a failure.

The Celonis Analytics Database should be backed-up on a regular basis, preferably to a remote location. You can use, for your reference, the Backup Policy already established by your IT Department or if such a thing is not available, you can set one up that will best suit your needs. This will allow the possibility of an Analytics Database recovery in case a disaster will occur.

When establishing a Celonis Analytics Database backup policy you must take into consideration at least the following topics:

- Database size
- Backup destination and available backup storage space

- The connectivity and thus the speed and throughput available from the Analytics Database Server to the Backup destination device or medium
- Database's high usage time frames
- Regular schedulers or Cron Jobs that were set-up together with our Data Scientist technical personnel during the Data Integration part of the Celonis installation

MONITORING THE APPLICATION SERVER

This section describes how to best monitor the Celonis Application Server in terms of resource usage to ensure that the application can run with maximum performance and efficiency.

It is best practice to constantly monitor the Celonis Application Server. Besides the initial minimum system requirements that are provided during the Celonis installation, additional resources must be always available, especially if the Celonis Application Server is sharing resources with other 3rd party software.

Unless specified otherwise, an operating system gets periodic updates that will increase its disk storage space necessity overtime. Additional disk storage space is also required so that the operating system can create periodic restore points. Extra disk storage space is to be considered if additional software, modules, libraries or features will also be installed on Celonis Application Server machine soon. These are just a few cases that will make you pay attention to disk storage space as other factors can influence this as well.

RAM and CPU resources are also very important. Insufficient RAM and/or CPU power may lead to very poor server performance, hang-ups or can even freeze entirely the Celonis Application Server. Most applications are making use of these two resources in a dynamic way (only when necessary), so it is very important to scale them properly.

Network throughput must be considered if the Celonis Application Server is shared with other 3rd party software that requires highly intensive and regular networking data transmission.

Taking care of everything at the start is quite easy, but this is not enough in a productive environment, especially in large IT infrastructures. In such cases (but not only) you should consider using (centralized) server monitoring techniques. There are a lot of tools and features that can provide you with real-time monitoring regarding all server resources, depending on the server's operating system and IT infrastructure. Having access to this kind of information in real-time will help you avoid unnecessary problems related to server overburdening.

INCLUDED MONITORING FUNCTIONALITY

JAVA MANAGEMENT EXTENSIONS

Apart from the Operating System's built-in monitoring capabilities, Celonis supports Java Management Extensions standard monitoring. To enable JMX, you should configure Celonis accordingly by adding the following Java properties on startup. Please note that the application needs to be restarted to activate those changes.

- -Dcom.sun.management.jmxremote
- -Dcom.sun.management.jmxremote.port=<port>
- -Dcom.sun.management.jmxremote.ssl=false
- -Dcom.sun.management.jmxremote.authenticate=false

The listed properties enable you to monitor the application remotely and unauthenticated on the configured Port "<port>" via HTTP. More information on JMX monitoring and advanced options/parameters (e.g. for setting up monitoring via HTTPS and using authentication) can be found in the Oracle Guide Monitoring and Management Using JMX Technology.

Additional java options are added to the "cpm.user.vmoptions" file as described here.

CELONIS MBEANS

Opening a JMX Console and connecting to the configured monitoring port already gives the possibility to check the RAM and CPU usage in real-time. Even more, the following MBeans are predefined, to provide the capability to monitor specific internal Celonis processes:

- "DataManagement" with the following attributes: "ActiveLoads", "LastFailedLoads", "LoadedDataModels", "Schedules", "SystemDataModels"
- Logging with the following attributes: "LogLevel"
- "SystemResources", with the following attributes: "CacheFreeMb", "CacheUsedMb", "CpuUsage", "RamAvailable", "RamInUse"
- "UserManagement" with the following attributes: "UserCount"

Each of the attributes can be used to interpret specific Celonis activities as following:

- "ActiveLoads": Which loads are active at the time
- "LastFailedLoads": Which loads have failed
- "LoadedDataModels": Which Data Models are loaded
- "Schedules": What schedules are active
- "SystemDataModels": The number of SystemDataModels
- "LogLevel": The Log level

- “CacheFreeMb”: The available RAM that can be used by the Cache.
- “CacheUsedMb”: How much RAM is used by the Cache.
- “CpuUsage”: CPU usage value. This is only available for Linux Operating Systems
- “RamAvailable”: The amount of available RAM
- “RamInUse”: The amount of RAM that is currently being used
- “UserCount”: How many users are currently logged in the application

WILY INTROSCOPE

The application can also be integrated to be monitored with Wily Introscope. More information on Wily Introscope and its setup can be found in SAP Note 797147.

To configure Celonis for Wily Introscope integration, add the following Java properties on startup using the same method as for JMX above.

- “-Dcom.wily.introscope.agent.agentName=<uniqueName>”
- “-javaagent:<wilyInstallDir>\Agent.jar”
- “-Dcom.wily.introscope.agentProfile=<wilyInstallDir>\core\config\IntroscopeAgent_tomcat.profile”
- “-XX:-UseSplitVerifier”

“<wilyInstallDir>” is the path where you installed/extracted the Wily Introscope Agent. There is no specific preconfigured agent profile for jetty, but you can reuse the tomcat profile. Please review SAP Note 1438005 regarding the installation procedure of the Introscope Java Agent for Apache Tomcat server. In the “IntroscopeAgent_tomcat.profile”, you need to configure at least the following properties, so that the agent will be able to find the enterprise manager installation:

- “introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=localhost”
- “introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=6001”

MONITORING THE ANALYTICS DATABASE

This section describes how to best monitor the analytics database in terms of resource usage to ensure that the application can run with maximum performance and efficiency.

The underlying analytics database should be monitored as well as with regards to average utilization, disk and memory space, or performance in general. A detailed description of monitoring queries and performance on your analytics database is out of scope of this Operations Guide. Please refer to the official documentation for your database system.

LOGGING AND TRACING

Learn about the log files that will be created during the usage of Celonis as well as about how to manage them.

You can always make use of the Celonis Application Server logging system. The log files generated by the Celonis Application Server are in the “logs” folder that resides in the root of the Celonis Application Server install path. The logs generated by the Compute Service are stored in the “compute” folder. These logs can offer you information related to:

- Starting and Stopping the Celonis Application Server
- Exceptions of Celonis
- Queries and their timing
- Other information related to Celonis Application Server

Celonis offers the possibility to configure different logging levels. The logging levels can be configured from the “config-custom.properties” file (see **logging.level** property). For this guide, the most significant information is that you can assign different levels to each package: “INFO”, “WARN”, “ERROR”, “DEBUG”. Please note that changes to the logging parameters will require a restart of the Celonis to take effect.

Note: The information gathered by each of the logging and tracing systems should be used only for debugging purposes.

SOFTWARE CHANGE MANAGEMENT

This section describes how changes to the software are managed. New releases and support packages can be retrieved from my.celonis.de. Regardless of the type of patch, you will be provided with a full installer file. The procedure for updating an installation is described in the next chapter Software Update Procedure.

When you want to promote configurations and artifacts to production, there is a built-in export/import mechanism for all transportable artifacts in the web interface of Celonis; for usage instructions, please refer to the Celonis manual. Technical configurations can be copied on a file level.

SOFTWARE UPDATE PROCEDURE FOR SERVICE PATCHES

For the installation of major releases (*4.x*), and minor releases (4.7.x), please always refer to the official Celonis Update Guide of the respective version, as additional steps might be required.

SUPPORT DESK MANAGEMENT

This part lists the contact details of the service desk as well as the best procedure for getting in contact with it in case of problems.

To contact Celonis support, you have the following possibilities:

Hotline: **+49 (0)89 416 159 677**

Email: servicedesk@celonis.de

Support-Portal: https://celopeers.com

Please include at least the following items in your issue description:

- At what time the incident occured (date, time, timezone)
- Screenshot or video recording of the error message / situation
- Full product version tag (format: *4.7.2-RELEASE_c3739d3_20211027_1623*) that can be found:
    - In the Celonis frontend, clicking "*About*" on the bottom left
    - In the log files
    - In the file *config.properties#application.versio*n inside the installation directory
    - In the file *FILEID* inside the installation directory
- Log files for the past 30 days:
    - Central Application logs: *logs/stdout, logs/stderr*
    - Compute logs: *compute/logs/stdout, compute/logs/stderr*
    - Query engine logs: *compute/root/logs*
- [Linux only] The content content of the following scripts in the installation directory
    - *start.sh*
    - *stop.sh*
    - *start_application.sh*
    - *stop_application.sh*
    - *compute/start.sh*
    - *compute/stop.sh*
- Installation location (e.g., C:\Program Files\Celonis 4 Enterprise or /opt/celonis/cpm4)
- Configuration files inside the installation directory:
    - All **.vmoptions* files
    - *config.properties*
    - *config-custom.properties* (anonymized sensitive information such as(passwords, certificate keys etc)
- Operating system information of the application server (name, version, kernel)

- Hardware information of the application server (available RAM, CPUs, persistent storage)

CONFIGURABLE HELP PAGES

It is possible to adjust the help pages displayed in Celonis for the users (either via <Username> -> Help or via an analysis -> Help) and the support contact on the login page. These settings can be found in the “config-custom.properties” section “Help page customization”.

TROUBLESHOOTING

Refer to this section to find a list of common issues and first instruction for solving them.

- Application not accessible:
    - Are you connected to the corporate network?
    - Do you use a proper (up-to-date) web browser? Supported browsers are Google Chrome (min. Version 40, preferred), Mozilla Firefox (min. Version 38) or Internet Explorer (min Version 11).
    - Is the URL you are trying to access correct?
    - Is the (Database) Server running?
    - Is Celonis Application Server running? (For your reference, you can check the " Celonis as Operating System Service" chapter of this guide)
    - Is(Are the) Compute Service(s) running?
    - Login failed:
        - Double-check that you have entered the correct password.
        - Double-check that you have entered the correct user name.
        - Does the User Account exist?
        - Note: Passwords in Celonis are case-sensitive.
- Analysis is empty, no data is showing
    - Are the permission rights set correctly?
        - For the Analysis?
        - For the Data Model?
    - Was the Analysis saved after modification?
    - Were all selections reset?
    - Are the permanent filters deactivated?
    - Is the database connection successful?
    - Have you checked for any errors in data integration?
- Document/Data Model disappeared?
    - Were the permissions withdrawn?
    - Was any restore performed with an older backup?
        - For your reference, you can check the Backup and recovery – Backup Analytics chapter of this guide.
- Transports not working on RHEL? This issue is related to the configuration of the Red Hat Operating System, which is cleaning files in the /tmp directory:
    - Edit the configuration file "/usr/lib/tmpfiles.d/tmp.conf" and add "x /tmp/jetty*" to exclude jetty* files being removed by the clean-up scheduler
    - Afterwards, restart the Central Application as well as the Compute Service(s). All data models have to be reloaded

For further information on troubleshooting, please also consult the troubleshooting section of https://help.celonis.de/.

REFERENCES

This part lists the reference details of the Celonis Operation Guide.

- Celonis Manual
- Oracle Manual
- Oracle Guide Monitoring and Management Using JMX Technology
- SAP Note 797147
- SAP Note 1438005