UPDATE GUIDE

Version 1.10

Corresponding Software Version

Celonis 4.6

TABLE OF CONTENTS

REVISION HISTORY

VERSION NUMBER	VERSION DATE	SUMMARY OF REVISIONS MADE
1.1	FEB 16, 2017	Application Version 4.2
1.6	FEB 23, 2018	Updated version for application version 4.3
1.7	MAI 05, 2018	Updated version for application version 4.4
1.9	MAR 03, 2019	Updated version for application version 4.5
1.10	DEC 03, 2019	Updated version for application version 4.6

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.

This guide provides a step-by-step instruction on how to update Celonis or apply a patch downloaded from my.celonis.de. An overview of the changes and new features can be found in the release notes.

TARGET AUDIENCE

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

- System Administrators
- Support Personnel
- Technical Staff

LIST OF ABBREVIATIONS

ABBREVIATION	EXPLANATION
ADFS	Active Directory Federation Service
ID	Identifier
OS	Operating System
SAML	Security Assertion Markup Language
SSO	Single Sign-On
URL	Uniform Resource Locator
ZIP	Zipper (Archive File Format)

SOFTWARE CHANGE MANAGEMENT

New releases and support packages are announced on my.celonis.de and can be retrieved from there. 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. For detailed installation instructions, please refer to the Celonis Installation Guide to find the latest application prerequisites. For detailed operating instructions, please refer to the Celonis Operation Guide to understand the Celonis application in depth.

Please note:

- A Celonis Patch is a resolution or fix for one specific issue
- A Celonis Service Pack resolves multiple issues
- Patches or Service Packs may be available in advance, if critical
- A Celonis Release is a new version of the software, including new features

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 (help.celonis.de). Technical configurations can be copied on a file level.

SOFTWARE UPDATE PROCEDURE

LATEST CELONIS VERSION

The Celonis software is shipped as an installer. The installer type depends on the Operating System it is going to be installed on. Verify the correct Celonis software version before you deploy any update. The latest Celonis software version can be downloaded from my.celonis.de.

The general update procedure is described below, however there may be several other instructions specific to a certain release. If any specific instructions should apply, they will be shipped out together with the release.

There will be a short downtime of the Celonis Application for the duration of steps 2 to 6.

MIGRATION FROM CELONIS PROCESS MINING BY CELONIS 4.5 TO 4.6

All new features can be found in the release notes for Celonis Process Mining 4.6. In the following section, migration and updates are described.

Enhanced application service design

With Celonis Process Mining 4.6, the Celonis Services on the application server are split into the Celonis Central Application Service and the Compute Service. This new Compute Service manages the resource-heavy engine processes (which load and subsequently hold all Data Model data and calculate and provide the response to PQL queries). The Celonis 4.6 Central Application Service handles all UI resources as well as the data integration, which includes the orchestration of Data Model loads.

Both applications, the Central Application and the Compute Service can be started and stopped separately. One of the big advantages of this new design is that Celonis 4.6 application configurations can be enabled and effected by restarting the Central Application Service and without affecting the load status of all Data Models. The Data Models remain loaded until they are manually unloaded or the Compute Service is stopped.

The default installation routine starts and stops both applications to provide a seamless transition from Celonis 4.5 to Celonis 4.6. For Windows users it is recommended to start all services in the service manager application.

After successful migration, you might see the following error message during Data Model loads. This is the error message for when the Compute Service has not been started yet.

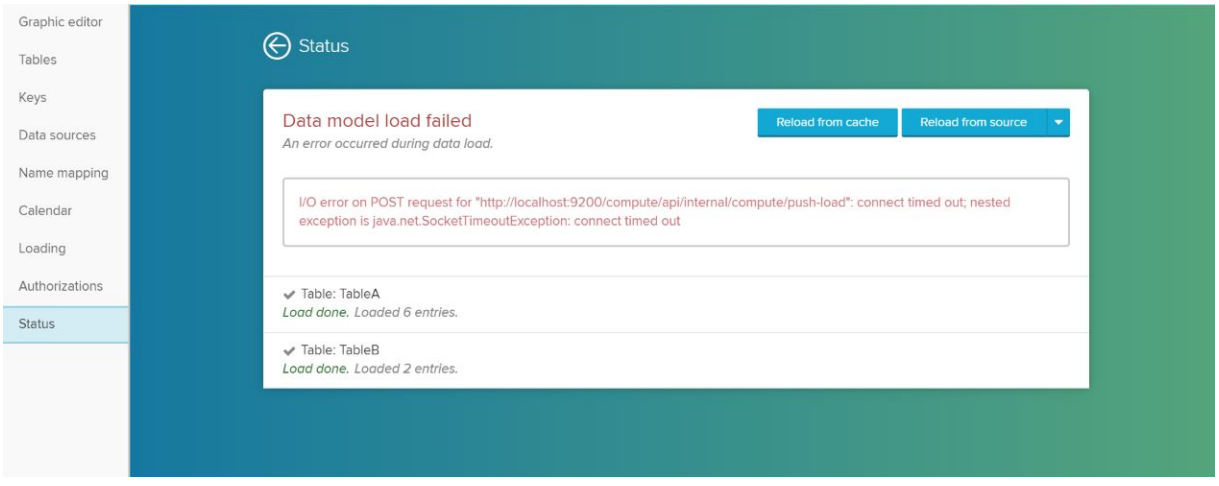

Figure 1: The compute service on port 9200 has not been started

Multi-Server Deployment

Since Celonis Process Mining 4.6, a Multi-Server Deployment is possible. It allows you to host the Compute Service on multiple servers. For this deployment, it is not necessary that the Compute Service is running on the Celonis Application Server. For future versions, every Compute Service needs to be updated independently. For a step-by-step guide on how to initially set up this architecture, please consult the Celonis Installation Guide as well as the Celonis Operation Guide.

JRE folder

Since Celonis Process Mining 4.6, the "jre" folder in the "Celonis 4 Enterprise" directory will be entirely deleted and recreated with every installation. Any non-system files inside this folder will be removed during this process. It is recommended not to modify any files other than the configuration files inside the Celonis installation folders.

Analytics Database

The default Compute Service uses embedded HSQLDB (http://hsqldb.org/) by default as the Analytics Database. PostgreSQL (https://www.postgresql.org/) > 9.6 and Microsoft SQL Server > 2017 are supported as Analytics Database. **H2 databases and PostgreSQL versions prior to 9.6 are not supported.**

INTEGRITY OF CELONIS CONFIGURATION FILES

Celonis updates may require changes on configuration files. The following configuration files exist:

- "config.properties"
- "config-custom.properties"
- "config-custom.properties.sample"
- "compute/application.properties"
- "compute/application-custom.properties"
- "compute/application-custom.properties.sample"

The configuration files "config.properties", "config-custom.properties.sample", "application.properties" and "application-custom.properties.sample" will be overwritten by the new version. Your custom configuration files "config-custom.properties" and "application-custom.properties" remain unchanged. Please check the new "config-custom.properties.sample" and "application-custom.properties.sample" files for changed and updated parameters.

SAML configuration

If you are utilizing Single Sign-On (SSO) with ADFS (SAML authentication), please check if the SAML configuration in the “config-custom.properties” file is up to date. The parameters “saml.url.scheme”, “saml.url.serverName”, “saml.url.serverPort”, and “server.url” must be added to your SAML configuration. For details on how to set these parameters refer to “Celonis ADFS Setup Guide 1.9”.

BACKUP OF THE CELONIS SERVICES AND CONFIGURATION STORE

Step 1: Stop the Celonis services

- Windows: Stop the services “Celonis CPM4” and “Celonis CPM4 compute” in the Windows Service Manager.
- Linux: Execute the “stop.sh” script to start the application service and the Compute Service.

In the case of the above-mentioned Multi-Server Deployment, every Compute Service has to be stopped and backed up individually.

Verify that the Celonis processes have been terminated successfully.

- Windows: Investigate the currently running processes using the Task Manager
- Linux: Investigate the currently running processes by the Linux Process Table

Step 2: Create a backup of the Celonis application and application data:

Create a backup of the following files and folders:

- <installPath>/component_configuration
- appfiles/appdata.* (on Windows), root/appdata.* (on Linux)
- <installPath>/compute/root/appdata.* (for every Compute Service)

Optional: Perform a file system level backup of the installation and appfiles folder using customer specific backup solution (e.g. Tivoli Storage Manager, Symantec/Veritas Backup Exec, etc.).

In case you are already using the Celonis Configuration Store on an external database system, create a backup of the Celonis Configuration Store.

In case you are using the integrated Celonis Configuration Store powered by HSQLDB, consider migrating the Celonis Configuration Store to an external database system. For more information, please refer to the Celonis Configuration Store Setup Guide.

Step 3: Backup additional files

Verify to have a recent backup of all files, which you might have modified or added additionally (e.g. HANA JDBC driver in appfiles/app/WEB-INF/lib (on Windows), root/app/WEB-INF/lib (on Linux)) as they may be overridden by an update.

GENERAL UPDATE PROCEDURE

Step 1: Download the new release from my.celonis.de

Download the correct installer for your host OS. If you cannot find the installer matching your OS and version, please contact the Celonis Servicedesk.

Step 2: Stop the Celonis services

- Windows: Stop the services “Celonis CPM4” and “Celonis CPM4 compute” in the Windows Service Manager.
- Linux: Execute the “stop.sh” script to stop the application service and the compute service.

In the case of a Multi-Server Deployment, every Compute Service has to be stopped (Linux: using the “stop_compute.sh” script) and then updated individually.

Verify that the Celonis processes have been terminated successfully:

- Windows: Investigate the currently running processes using the Task Manager
- Linux: Investigate the currently running process by the Linux Process Table

Make sure to close any other applications afterwards (esp. the Windows Service Manager on Windows and any Windows Explorer windows or Linux command line locations inside the installation/application path).

Step 2a: Rotate logfiles (Linux only – optional)

In general there are two options: manual or with logrotate. In order to set up log rotate it is not necessary to stop the Celonis service.

Option 1: Manually

On Linux systems, there is no automatic log file rotation for the files stdout and stderr. If you use manual rotation by moving the files stderr and stdout to e.g. stderr.0 and stdout.0 respectively, such that after the update the files stderr and stdout can be written to from scratch. Consider compressing the rotated files (stderr.0, stdout.0). In case such a rotated file already exists, consider deleting or renaming these old log files.

Option 2: Logrotate

For information on how to set up Logrotate, please consult the chapter “Celonis Log Files” in the Celonis Operations Guide.

Step 3: Run the Celonis Installer

On Windows, the installer will recognize your current setup and keep your system configuration settings. The installer will automatically update Celonis to the latest version.

Continue only if the installation finished successfully.

On Linux, you will have to re-enter the initial configuration parameters even for an update.

Step 3a (Multi-Server Deployment only): Update the Compute Services individually

- **Windows**
    1. Execute “compute_svc.exe uninstall” inside of the “compute” folder as an administrator on the Celonis Compute Server.
    2. Copy the folder “jre” as well as the folder “compute” from the install directory of the Central Application into a **newly created** directory on the respective Compute Server.
    3. Copy the **“application-custom.properties”, the “log” folder, the “root” folder, and the “temp” folder** from the previous installation directory of the Compute Service into the newly-created directory. This step ensures that custom configurations are preserved.
    4. Copy the files “vcredist_2015_x64.exe”, “vcredist_2010_x64.exe” and “vcredist_2008_x86.exe” from the install directory to the Celonis Compute Server and execute each of them.
    5. Execute “compute_svc.exe install” inside of the **newly created** “compute” folder as an administrator on the Celonis Compute Server.
- **Linux**
    1. Copy the folder “jre” as well as the folder “compute” from the install directory of the Central Application into a **newly created** directory on the respective Compute Server.
    2. Copy the “**application-custom.properties”, the “log” folder, the “root” folder, and the “temp” folder** from the previous installation directory of the Compute Service into the newly-created directory. This step ensures that custom configurations are preserved.

Step 4: Start the Celonis services

- Windows: Start the “Celonis CPM4” service for the Central Application and the “Celonis CPM4 compute” service for the Compute Service from the Windows Service Manager. Additionally the Compute Service of every separate Compute Server has to be started as well.
- Linux: Execute the “start.sh” script to start the Central Application service and the local Compute Service. Execute the “start_compute.sh” to start the Compute Service on separated Computer Servers.

Verify that the Celonis processes have been started successfully:

- Windows: Investigate the currently running processes using the Task Manager
- Linux: Investigate the currently running process by the Linux Process Table

Step 5: Verify that logs files are written

Login to the Celonis application. Verify if the log files are written:

- Windows: Log-files are separated per service start and can be found in “<installPath>\logs” and “<installPath>\compute\logs”.
- Linux: Log-files are combined in “<installPath>/logs” and “<installPath>/compute/logs”.

Step 6: Verify the Celonis release ID

Login to the Celonis application and access the “About” page in the bottom left corner of the start screen. Validate that the displayed version ID equals the new release.

If you want to check the version of Celonis while the software is not running, you can do so by viewing the “config.properties” file in the root directory of the Celonis application (parameter “application.version”).

PERFORM CLEAN-UP OPERATIONS

1. We recommend keeping at least the latest backup archive
2. Delete obsolete backup archives
3. Verify and update the integrity of Celonis configuration files

POST UPDATE STEPS – QUERY CACHING

In order to ensure that data models and analysis benefit from the performance improvements that is embedded in Celonis Process Mining 4.6 make sure that query caching is enabled for larger data models (recommendation: > 5 Mio cases).

By enabling query caching, long running queries whose execution time exceeds one second, will be executed and cached while loading the actual data model. Multiple executions of a query are accelerated as well. Note that you must reload the data model for the setting to become active. Please note, the query cache allocates memory in the according heap of the application server.

Best Practices

- Query cache size: 150MB
- Warm Up Time: 15-30 minutes

The setting can be enabled in the data model under *Loading* (see Figure 2).

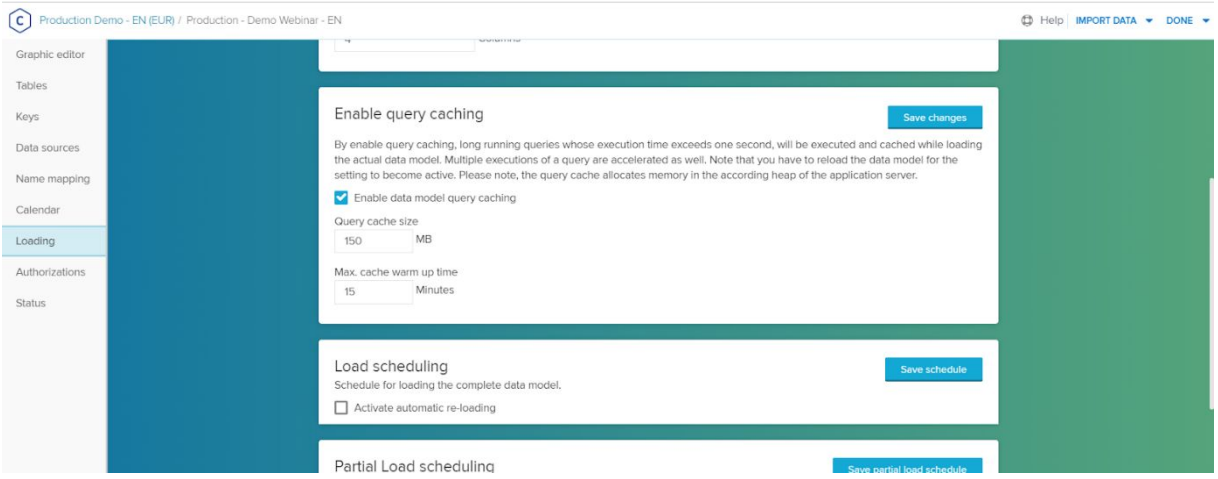

Figure 3: Query Caching Settings in Data Model

MIGRATION FROM CELONIS PROCESS MINING FROM A VERSION BELOW 4.5 TO A VERSION INCLUDING AND ABOVE 4.5

With Celonis 4.5 we included a performance improvement for the whole application which requires a reload from source for each Data Model. The initial load can take longer than expected.

For the migration from the older version Celonis 4.3 to Celonis 4.6, please first upgrade to Celonis 4.5. During this upgrade, additional steps have to be taken into consideration.

Cycle Joins in data models

A new feature of the Celonis Data Model will support you in building fully functional data models. Therefore, we block data models with cycle joins during the load from Celonis Process Mining 4.5 on and users will get a warning during data model load including the corresponding tables that embed the cycle join (see Figure 1). If you have Python for Celonis installed for Celonis Process Mining 4.5 then it is possible to detect cycle joins with a python script that can be found in the Appendix. The script will provide a list with data models that have cycle joins and the user responsible for the data model should then resolve the cycle joins in data models.

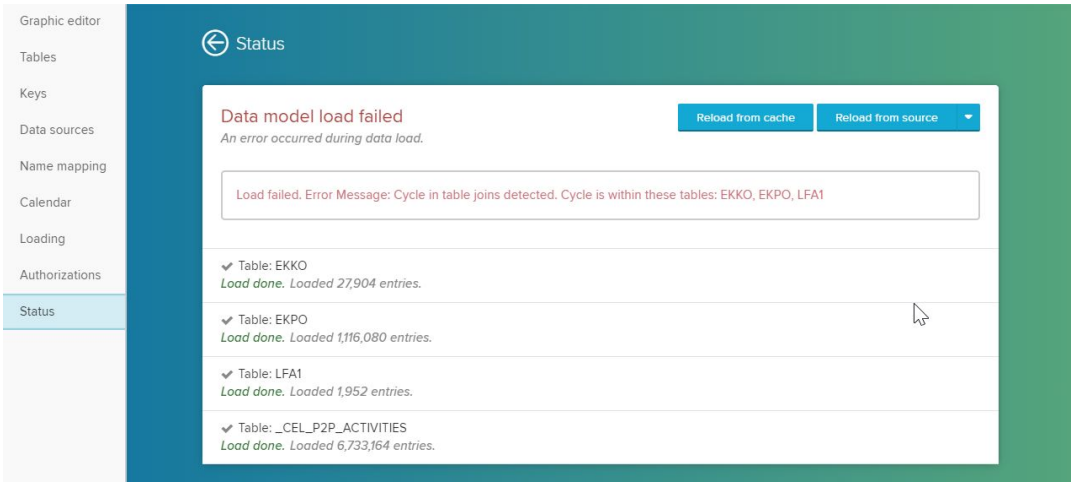

Figure 2: Cycle Join in Data Model

When there are cycles in the join information, the joins are not uniquely defined. Apart from that, the Cycle Joins should be avoided for the following reasons:

- PullUp functions can lead to false calculations in your analysis
- Compute Node/Query Engine can be shut down due to cycle joins and this will influence all other running data models on your productive environment
- Data Model quality needs to be guaranteed for all users and cycle joins are not a good practice (especially loading times of data model can take longer due to cycle joins)

Bookmarks

Bookmarks are saved in a new format in Celonis Process Mining 4.5. We will migrate all bookmarks automatically. We still recommend backing up and extracting all existing bookmarks to ensure you can recover them if a migration fails.

Load Scripts

Load scripts on analysis level are improved in Celonis Process Mining 4.5. We will migrate all load scripts automatically. We still recommend backing up and extracting all existing load scripts to ensure you can recover them if a migration fails.

PQL Statements: SELECT | CLEAR SELECTIONS | SELECT PINNED

Up to Celonis Process Mining 4.4 the PQL statements *SELECT, CLEAR SELECTIONS,* and *SELECT PINNED* have been supported. With Celonis Process Mining 4.5 this PQL Statements are not supported anymore and are migrated automatically:

- ***SELECT PINNED*** statements in load scripts can not be used anymore in the Analysis UI. As substitute the functionality ***Publish with Selections*** and ***Restore default selections*** is included.
- ***SELECT & CLEAR SELECTIONS*** can not be used in load scripts (or in other formula editors) and is migrated to ***FILTER.*** In order to apply selections in load scripts FILTER is the only available functionality that filters the data.
- ***SELECT "EVENTLOG"."TABLE" IDS [1, 2, 3, 4, 6, 7, 11, 12, 14, 22, 23, 24, 25]*** or any other dynamic filters are not migrated and do not work as intended as FILTER is the only applicable statement for load script statement.

SUPPORT DESK MANAGEMENT

To contact Celonis support, you have the following possibilities:

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

Email: servicedesk@celonis.de

Support-Portal: https://servicedesk.celonis.com

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

- Used browser including version (e.g. Google Chrome Version 64.0)
- Installation which you are trying to access (in case there is e.g. Dev and Prod)
- URL used to access the system (sometimes, there can be more than one URL to reach a single installation. This will also help to identify the installation you are trying to access)
- User name used to logon
- Screenshot of the error message/situation
- Log files of the application (if accessible on the server)

For additional information please refer to “Service Description For Celonis Support Services” available on the official Celonis website.

REFERENCES

- Celonis Installation Guide
- Celonis Operation Guide
- Celonis Manual
- Celonis Release Notes
- Celonis ADFS Setup Guide 1.6
- Celonis Migration Store Setup Guide

APPENDIX

PYTHON SCRIPT TO DETECT CYCLE JOINS

A new feature of the Celonis Data Model will support you in building fully functional data models. Therefore, we block data models with cycle joins during the load from Celonis Process Mining 4.5 on. Prerequisite is that Python tools are installed for Celonis Process Mining 4.4. The script should be run before upgrading to Celonis Process Mining 4.5.

```
1. ##########################################################################################
   ##
2. ##########################################################################################
   ##
3. ############################ SET YOUR CONFIGURATION HERE ############################
   ##
4. ##########################################################################################
   ##
5. ##########################################################################################
   ##
6.
7.
8. BASE_URL="http://localhost:9000/cpm/"
9. ADMIN_USER="sysadmin"
10. ADMIN_API_TOKEN="INSERT TOKEN HERE"
11. ADMIN_API_SECRET="INSERT SECRET HERE"
12.
13.
14. ##########################################################################################
   ##
15. ##########################################################################################
   ##
16. ######################## DO NOT CHANGE ANYTHING AFTER THIS LINE ####################
   ##
17. ##########################################################################################
   ##
18. ##########################################################################################
   ##
19.
20.
21. import celonis_tools
22. from celonis_tools.api import CelonisSession
23. from celonis_tools.model import DataModel
24. from collections import defaultdict
25.
26. class Graph:
27.     def __init__(self):
```

```
        self.graph = defaultdict(list)

    def add_edge(self, v, w):
        self.graph[v].append(w)
        self.graph[w].append(v)

    def is_cyclic(self, start_point):
        #adapted from http://darrencheng.coding.me/2016/03/11/Algorithm/Detect-cycle-in-undirected-graph/
        queue = [start_point]
        visited = set()
        visited.add(queue[0])
        while queue:
            from_point = queue.pop(0)
            for to_point in self.graph[from_point]:
                if to_point in visited:
                    return True
                else:
                    queue.append(to_point)
                    visited.add(to_point)
                self.graph[to_point].remove(from_point)
        return visited


def get_unique_fks(dm):
    fk_hashes = set()
    fks=[]
    for fk in dm.foreign_keys:
        ids = [fk.tables[0].id, fk.tables[1].id]
        ids.extend([c1.id for (c1,c2) in fk.columns])
        ids.extend([c2.id for (c1, c2) in fk.columns])

        ids.sort()

        fk_hash = "_".join(ids)
        if fk_hash in fk_hashes:
            #duplicate key
            print("Duplicate key found in DataModel {} ({}): {}-{}".format(dm.id, dm.name, fk.tables[0].name, fk.tables[1].name))
        else:
            fks.append(fk)
            fk_hashes.add(fk_hash)

    return fks

with CelonisSession(base_url=BASE_URL, username=ADMIN_USER, api_token=ADMIN_API_TOKEN, api_secret=ADMIN_API_SECRET) as session:
    dataModels = DataModel.load_all()

```

```
    cyclic_data_models = []
    duplicate_fks_data_models = []
    for dm in dataModels:
        print("\nScanning in DataModel {} ({}) in project '{}'".format(dm.id, dm.nam
e, dm.folder.parent.name))
        fks = get_unique_fks(dm)

        if len(fks) != len(dm.foreign_keys):
            duplicate_fks_data_models.append(dm)
        for table in dm.tables:
            g = Graph()
            for fk in fks:
                g.add_edge(fk.tables[0].id, fk.tables[1].id)

            cyclic = g.is_cyclic(table.id)
            if(isinstance(cyclic, set)):
                #no cycles
                if len(cyclic) == len(dm.tables):
                    # all tables visited. No cycle!
                    break
                else:
                    # disconnected graph, try again with other start point
                    continue
            else:
                #cycle
                print("Cycle found in DataModel {} ({})!".format(dm.id, dm.name))
                cyclic_data_models.append(dm)
                   break

    print("\nSummary:")
    if cyclic_data_models:
        print("The following datamodels contain cyclic foreign keys:")
        [print("    * DataModel {} ({}) in project '{}'".format(dm.id, dm.name,
dm.folder.parent.name)) for dm in cyclic_data_models]
    else:
        print("No datamodels with cyclic joins found.")
    if duplicate_fks_data_models:
        print("The following datamodels contain duplicate foreign keys:")
        [print("    * DataModel {} ({}) in project '{}'".format(dm.id, dm.name,
dm.folder.parent.name)) for dm in duplicate_fks_data_models]
    else:
        print("No datamodels with duplicate joins found.")
```