

## ConSol CM Operations Manual

# Table of Contents

---

1	Introduction	6
1.1	ConSol CM	7
1.2	ConSol CM Documents	8
1.3	System Overview	9
1.4	The Book's Structure	10
1.5	Conventions Used In This Book	12
1.6	Legal notice	13
1.7	Gender Disclaimer	14
1.8	Copyright	15
2	ConSol CM Operations Manual - System Architecture	16
2.1	System Architecture	17
2.2	System Overview	18
2.2.1	System Architecture	18
2.2.2	LDAP Authentication	22
2.3	Architecture of a CM System with CMRF and DWH	23
2.3.1	Overview	23
2.3.2	DWH Database	26
2.4	Architecture of the ConSol CM Application	27
2.4.1	Introduction	27
3	ConSol CM Operations Manual - File System Structure	28
3.1	ConSol CM File System Structure	29
3.1.1	Server	29
3.1.2	Client	36
3.1.3	Variables Used for Standard Path Values in this Manual	36
3.2	ConSol CM Logging and Log Files	37
3.2.1	Introduction	37
3.2.2	Log Files	37
3.2.3	Using Log Levels for Troubleshooting	40
3.2.4	Using Logging in Scripts	40
3.2.5	Configuring the Log Files Using Log4j (JBoss 5, Weblogic)	41
3.2.6	Configuring the Log Files in JBoss 7 (EAP 6.x)	44
4	System Startup and Shutdown of the CM Application Server	47
4.1	Introduction	48
4.2	JBoss 5	49
4.2.1	Windows	49
4.2.2	Ubuntu Linux	49
4.3	JBoss 7	51
4.3.1	Windows	51
4.3.2	Linux	51
4.3.3	General Config	52
4.4	WebLogic 11	53
4.4.1	Admin Server	53

4.4.2	Node Manager	55
4.4.3	Start Managed CM6 Server via WebLogic Administration Console	57
5	Data Backup, Restore, and Recovery Procedures	60
5.1	Introduction	61
5.2	Backup and Restore of the ConSol CM Configuration	62
5.3	Index Backup and Restore	63
5.4	(Backup and) Restore of a Completely Damaged DWH Database	64
6	System Update (Minor Version)	65
6.1	Introduction: Types of CM Installations	66
6.2	Updating a ConSol CM6 Standard Installation (for Case 1)	67
6.2.1	Information about the Update	67
6.2.2	Getting the Required Files	67
6.2.3	Perform the Update	68
7	Management of E-Mail Functionalities	69
7.1	Introduction	70
7.2	Short Overview of CM Components Relevant for Mailing	71
7.2.1	E-Mail Has to Be Fetched	71
7.2.2	E-Mail Has to Be Sent	72
7.3	Fine-Tuning CM Mailing	73
7.3.1	Changing Administrator E-Mail Addresses	73
7.3.2	Changing Mailing Parameters	73
7.4	Monitoring E-Mail Functionalities	74
8	Indexer Management	75
8.1	Introduction	76
8.2	Indexer Services	77
8.3	Indexer Directory Structure	78
8.4	Indexer Architecture and Update Principle	79
8.4.1	Basic Principle	79
8.4.2	Indexer Update, Step 1: Persistent Store Is Filled	80
8.4.3	Indexer Update, Step 2: Master Server Update	80
8.4.4	Administrative Changes	80
8.4.5	Indexer Update, Step 3: Slave Server Synchronization	81
8.4.6	Indexer Info about Manual Index Updates (Using the Admin Tool)	81
8.5	Indexer Restart	82
8.6	Index Update Failure Scenarios	83
8.6.1	Master Failover	83
8.6.2	Failure of Update Task	83
8.7	Monitoring the Indexer	84
8.8	Index Backup and Restore	85
8.8.1	Index Backup	85
8.8.2	Index Restore	85
8.9	System Properties Which Are Relevant for the Indexer Master/Slave Synchronization	87
8.10	JBoss Parameters Which Are Relevant for the Indexer Master/Slave Synchronization	88
9	ConSol CM Operations Manual - DWH Management	89
9.1	DWH Management	90
9.1.1	Introduction	90

9.1.2	Short Overview of CM Components Relevant for Reporting with CM	90
9.2	CMDB / CMRF/ Data Warehouse Synchronization Process	93
9.2.1	Introduction	94
9.2.2	CM / CMRF / DWH Synchronization: General Principle	94
9.2.3	CM / CMRF / DWH Synchronization: FAQs and Tips for Troubleshooting	105
10	ConSol CM and LDAP Authentication	108
10.1	Introduction to ConSol CM with LDAP	109
10.2	LDAP for Engineer Authentication in the Web Client	110
10.3	LDAP for Customer Authentication in the Portal CM.Track	111
11	ConSol CM License	112
11.1	General Information about Licenses in ConSol CM	113
11.1.1	Control of Consumed Licenses	114
11.2	Managing the ConSol CM License Using the Admin Tool	117
11.3	Expert Information about Accessing Content of CM Licenses	119
12	System Monitoring	121
12.1	Introduction	122
12.2	Monitoring Tools	123
12.3	Monitoring the ConSol CM Server Process	124
12.3.1	JMX Monitoring Using Jolokia	124
12.4	Monitoring the Login into the CM Clients (CM Web Client and CM.Track), End-to-End Tests	125
12.4.1	Monitoring User Configuration	125
12.4.2	Web Client Monitoring Principle	125
12.4.3	Monitoring CM in a Cluster	126
12.4.4	URL /logout for Automation Purposes	126
12.5	Using Beans to Monitor Some Basic Parameters	127
12.6	Checking the Status of the Indexer	128
12.6.1	Checking the Indexer File System	128
12.6.2	Checking the Indexer Status in the Database	128
12.7	Log File Monitoring	129
12.7.1	Tags to Monitor	129
12.8	Monitoring E-Mail Functionalities	130
12.8.1	CM Error E-Mail Configuration	130
12.8.2	Log File Control	130
12.8.3	Control of Undeliverable E-Mails	130
12.9	CMRF/DWH Monitoring	132
13	ConSol CM Release Management Process: Test - Staging - Production	133
13.1	Introduction	134
13.2	Deployment Pipeline	135
13.2.1	Development Server (DEV)	135
13.2.2	Test Server (TEST)	136
13.2.3	Staging Server (STAGE)	137
13.2.4	Production Server (PROD)	137
14	Troubleshooting FAQs	139
14.1	Introduction	140
14.2	General Checklist	141
14.3	What to Provide for the CM Support Team	142

14.4	Some Error Scenarios and Their Solution	143
14.4.1	Problems with Mailing	143
14.4.2	Problems with LDAP Authentication	144
14.4.3	Problems with Kerberos Authentication	144
14.4.4	Problems with CMRF/DWH	144
14.4.5	Problems with Web Clients (End Users: Engineers, Customers)	145
14.4.6	Problems with the Search / the Indexer	145
15	System (Fine) Tuning	147
15.1	Java Tuning	148
15.1.1	Start the ConSol CM Server with More RAM Space	148
15.2	CM Tuning	152
15.2.1	Change Duration of Web Client Sessions	152
15.2.2	Mailing	152
15.2.3	Search /Indexer	153
15.2.4	Database-Related Parameters	153
16	Running CM in a Cluster	155
16.1	Introduction	156
16.2	Basic Tips for Troubleshooting in a CM Cluster	158
16.2.1	Node Lost in Cluster	158
17	Appendix A - Glossary	159
18	Appendix B - Trademarks	167
19	Index	168

# 1 Introduction

---

- [ConSol CM](#)
- [ConSol CM Documents](#)
- [System Overview](#)
- [The Book's Structure](#)
- [Conventions Used In This Book](#)
- [Legal notice](#)
- [Gender Disclaimer](#)
- [Copyright](#)

## 1.1 ConSol CM

---



ConSol CM is a **customer centric business process management software**. Using ConSol CM you can control and steer business processes with a strong focus on human communication and interaction, e.g., user help desk, customer service processes, marketing and sales, or ordering processes. Basically, every process that is in operation in a company can be modeled and brought to life with ConSol CM.

When you read this manual, your company is presumably using ConSol CM as a process management application and it is your job to run and maintain the system. In this book you will learn how to operate a ConSol CM system from a technical point of view, e.g., you will get to know the system architecture, the log file structure and learn how to restart a ConSol CM server.

## 1.2 ConSol CM Documents

---

ConSol CM provides documentation for several groups of users. The following documents are available:

- **Administrator Manual**  
A detailed manual for CM administrators about the ConSol CM configuration using the Admin Tool.
- **Process Designer Manual**  
A guideline for workflow developers about the graphical user interface of the Process Designer and how to program workflow scripts.
- **Operations Manual**  
A description of the ConSol CM infrastructure, the server integration into IT environments and the operation of the CM system, for IT administrators and operators.
- **Set-Up Manual**  
A technical description for CM set-up in different IT environments. For expert CM administrators.
- **User Manual**  
An introduction to the ConSol CM Web Client for end users.
- **System Requirements**  
List of all requirements that have to be met to install ConSol CM, for IT administrators and CM administrators. Published for each ConSol CM version.
- **Technical Release Notes**  
Technical information about the new ConSol CM features. For CM administrators and key users. Published for each ConSol CM version.

For you as a CM system administrator the book you are reading, the *ConSol CM Operations Manual*, provides the required information to maintain a CM system. We assume that your company already has a ConSol CM system that is up and running. If you need information about the preparation and set-up of the system components, please refer to the *ConSol CM Set-Up Manual*. Most of the basic principles of the application are explained in the *ConSol CM Administrator Manual*.

You need support for ConSol CM operation? Call our support team.





## 1.3 System Overview

---

ConSol CM is a Java EE application which can run in standard application servers. It is released for JBoss and for Oracle Weblogic. As RDBMS (Relational Database Management System), Oracle, Microsoft SQL Server, or MySQL can be used.

For a detailed description of the supported operating systems, middleware, and other infrastructure components, please read the *ConSol CM System Requirements* of your ConSol CM version.

## 1.4 The Book's Structure

---

Since you can only operate a software when you have a profound knowledge of the technical background, the book starts with the section [System Architecture](#). Here, you will learn a lot about all components which form a functioning ConSol CM system and some interesting facts about the architecture of the application itself.

In the section [ConSol CM File System Structure](#), the most important directories and files are explained. Here you learn, for example, where to find the log files.

In the section [System Startup and Shutdown of the CM Application Server](#) you see how you can start, stop, and restart your ConSol CM system.

The section [Data Backup, Restore, and Recovery Procedures](#) tells you which databases, files, and directories have to be included into the every-day backup and how you can restore a broken ConSol CM system.

The section [System Update \(Minor Version\)](#) explains how to update a ConSol CM system. If you have a standard ConSol CM configuration, you can perform this step yourself. However, if you employ a ConSol CM version which has been built (developed) specifically for your company, this step can only be performed by your ConSol CM Consultant.

The management of the CM subsystems [E-Mail](#) (relevant for incoming and outgoing e-mails) and [Indexer](#) (relevant for search in CM) are treated in the two following sections.

If your CM systems is operated with a reporting infrastructure, you might refer to section [DWH Management](#) for information about DWH (Data Warehouse) and CMRF (CM Reporting Framework).

Everything you need to know about the ConSol CM license of your system is explained in the section [ConSol CM License](#).

The section [System Monitoring](#) describes how you can include your ConSol CM system into the company's monitoring concept.

Usually, a complete ConSol CM environment does not consist of a single server, but a complete system comprises a test and a production system, maybe even a staging system. The section [ConSol CM Release Management Process: Test - Staging - Production](#) describes the optimal way how to operate a system which runs in a productive system but which is also constantly improved and extended.

No operating software system without trouble. Unfortunately that's the way it is as you will know well from your every day business life. So this book also contains the section [Troubleshooting FAQs](#), where you hopefully find some support in case the system does not run as expected.

Useful hints and tips to optimize the CM system can be found in section [System \(Fine\) Tuning](#).

Since ConSol CM can not only be operated as single instance but also in a cluster, section [Running CM in a Cluster](#) explains the details about this topic.

You can look up all abbreviations and specific terms in the [Appendix A - Glossary](#), and please refer to the [Appendix B - Trademarks](#) for all complete product/software descriptions, including trademarks.

## 1.5 Conventions Used In This Book

---

Following are conventions used in this manual.

**Information:**

This is important additional information.

**Attention:**

This is an important note. Be careful here!

**Warning:**

This is a warning!

**Tip:**

This is a recommendation from our in-the-field consultants.

## 1.6 Legal notice

---

Since we would like to provide an administrator manual for you which helps you manage your CM system, but which also provides additional information about connected topics (e.g., LDAP, Kerberos), we have inserted external links into the manual. In this way, you can get some background information about a topic if you like. This can help you better understand the required CM configuration.

Despite careful review, we assume no liability for the content of those external links. The operators of sites linked to are exclusively responsible for their content.

## 1.7 Gender Disclaimer

---

As far as possible, ConSol CM manuals are written gender-neutral and often address the user with "you". When the phrasing "The user .... he ..." is used, this is always to be considered to refer to both, the feminine as well as the masculine form.

## 1.8 Copyright

---

© 2016 ConSol Consulting & Solutions Software GmbH - All Rights are reserved.

## **2 ConSol CM Operations Manual - System Architecture**

---



## 2.1 System Architecture

---

Here, you will learn a lot about all components which form a functioning ConSol CM system and some interesting facts about the architecture of the application itself.

Topic	Chapter
ConSol CM System	<a href="#">System Overview</a>
ConSol CM System with Reporting Infrastructure	<a href="#">Architecture of a CM System with CMRF and DWH</a>
ConSol CM Application	<a href="#">Architecture of the ConSol CM Application</a>

## 2.2 System Overview

---

- [System Architecture](#)
  - [Introduction to ConSol CM System Architecture](#)
  - [Basic System Architecture](#)
    - [CM Database](#)
      - [Oracle](#)
      - [Microsoft SQL](#)
      - [MySQL](#)
    - [Components for E-Mail Interactions](#)
    - [System Architecture with Reporting Infrastructure](#)
    - [Indexer](#)
- [LDAP Authentication](#)

### 2.2.1 System Architecture

#### Introduction to ConSol CM System Architecture



ConSol CM is a *Java EE* (Java Enterprise Edition) application that can be run in a standard application server on Unix/Linux or Windows systems. JBoss and Oracle WebLogic are supported.

A detailed list of supported application servers, database systems, and other systems is given in the current *System Requirements*.

In this chapter, a short overview of the ConSol CM system architecture will be provided.

## Basic System Architecture

ConSol CM is a Java EE application which is based on the classical three-tier architecture. The ConSol CM server is deployed in an application server and accesses a relational database. Two web interfaces are available as client interfaces. The standard interface is the ConSol CM Web Client which is used by the engineers to work on the tickets. Another web client is the ConSol CM portal, *CM.Track*. This provides access to the system for customers who might want to know some basic facts about the status of their tickets. The two Java applications which are used to configure ConSol CM are the *Admin Tool* and the *Process Designer*. Both can be downloaded from the ConSol CM start page using Java Web Start (JWS). For a couple of years now, JWS has been part of every Java edition, so no specific download is required on the PCs or Laptops you want to use to administer the system. On the contrary - you can do this from every regular web client with a supported web browser. Please make sure that the versions of all components which are used in your company meet the system requirements.

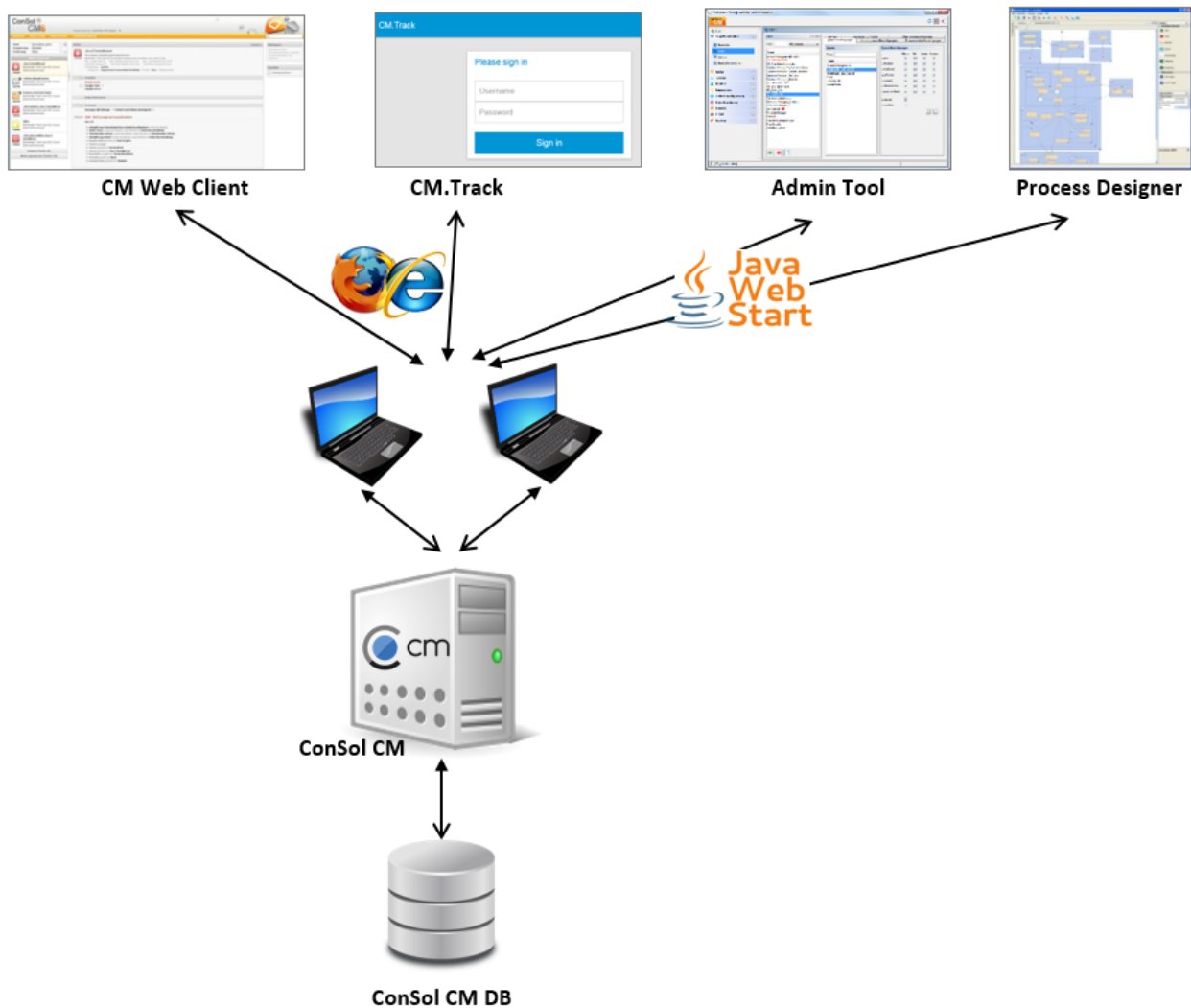


Fig. 1: ConSol CM - Basic System Architecture

## CM Database

The ConSol CM Database (CM DB) is a relational database which can be operated as Oracle, Microsoft SQL Server, or MySQL system (please see *ConSol CM System Requirements* for details).

### Oracle

One database schema with one database user is used by CM.

### Microsoft SQL

One database schema with one database user is used by CM.

### MySQL

One database with one database user is used by CM.

## Components for E-Mail Interactions

One of the core functionalities of ConSol CM is the integration with mail servers. This allows CM to send and to receive e-mails. For the engineer this means, new tickets can easily be opened via e-mail and the entire communication about a case is located in the respective ticket, including all incoming and outgoing e-mails.

In order to receive e-mails, ConSol CM connects to a mail server and retrieves e-mails from one or more mailbox(es). CM reacts like a regular e-mail client (e.g., Thunderbird, Microsoft Outlook) and uses standard e-mail protocols like IMAP or POP3. In case you want to use the secure version, IMAPS and POPS are also supported. For a detailed explanation about how to send encrypted e-mails with CM, please refer to the *ConSol CM Administrator Manual*, section *E-Mail Encryption*.



Please note that you might be dealing with different types of certificates here.

A certificate which is required to establish the connection between CM and an e-mail server usually has to be placed in the key store of the application sever.

Certificates which are required to encrypt e-mails which are written by the CM engineers using the Web Client have to be managed using the Admin Tool. This is explained in great detail in the *ConSol CM Administrator Manual*, section *E-Mail Encryption*.

In order to send e-mails, ConSol CM uses an SMTP server.

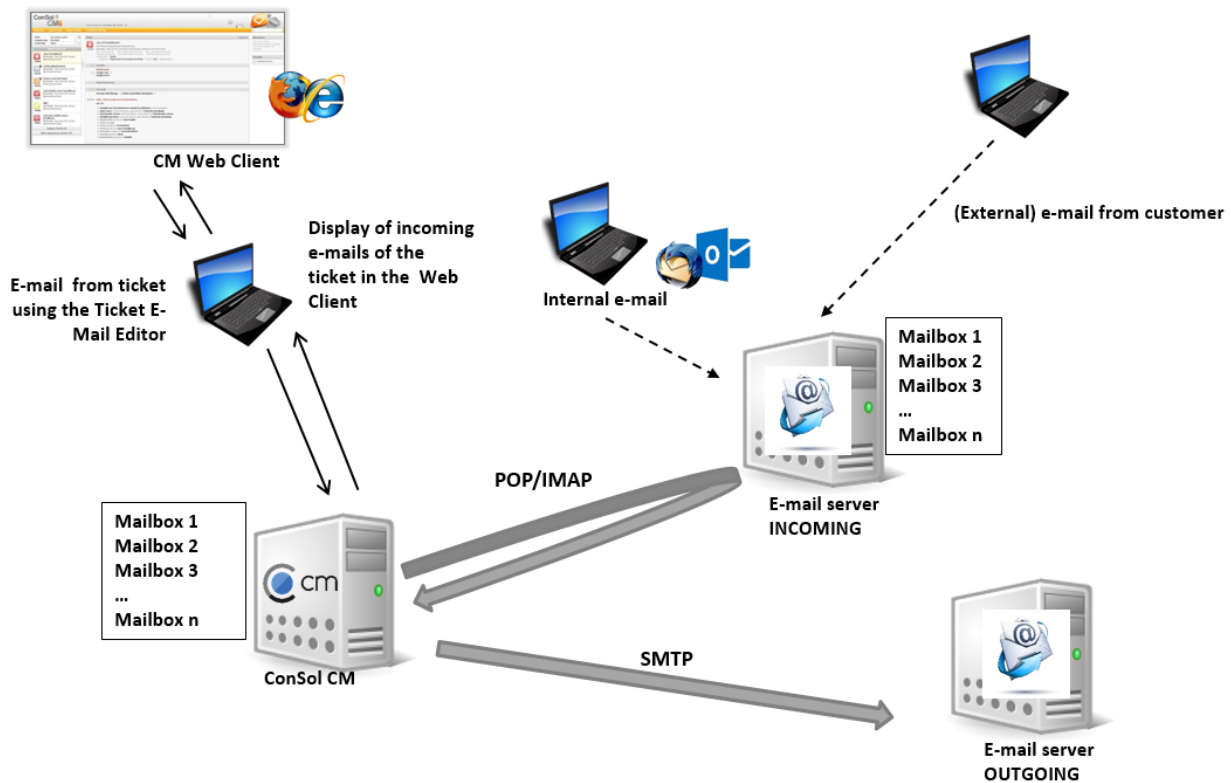


Fig. 2: ConSol CM - E-Mail Server Interactions

Please see also the explanations in the section [Management of E-Mail Functionalities](#).

## System Architecture with Reporting Infrastructure

This is explained in section [Architecture of a CM System with CMRF and DWH](#).

## Indexer

In order to perform effective searches in the database, CM builds an index for each Custom Field, Data Object Group Field, and Resource Group Field which should be included in a search. Furthermore, the engineer data, the ticket comments, and the attachments are indexed per default. The indices are stored on the file system! Please refer to the section about the data directory for an explanation of the index directory structure and read the detailed introduction to the entire topic in the *ConSol CM Administrator Manual*, section *Search and Indexer Configuration*. The section [Indexer Management](#) of the current manual treats the topic from a system operator's point of view.

## 2.2.2 LDAP Authentication

As standard feature, ConSol CM can use LDAP authentication in the Web Client and/or in the portal (CM. Track). Depending on the configuration of your LDAP server (e.g., Microsoft Active Directory), a user name and password might be required to establish the LDAP connection. All LDAP parameters are stored as ConSol CM system properties. Please read the detailed description in the *ConSol CM Administrator Manual*, sections *LDAP Authentication in the Web Client* and *CM.Track (V1/V2) - Authentication Modes for the Portal* in case you need to change the settings.

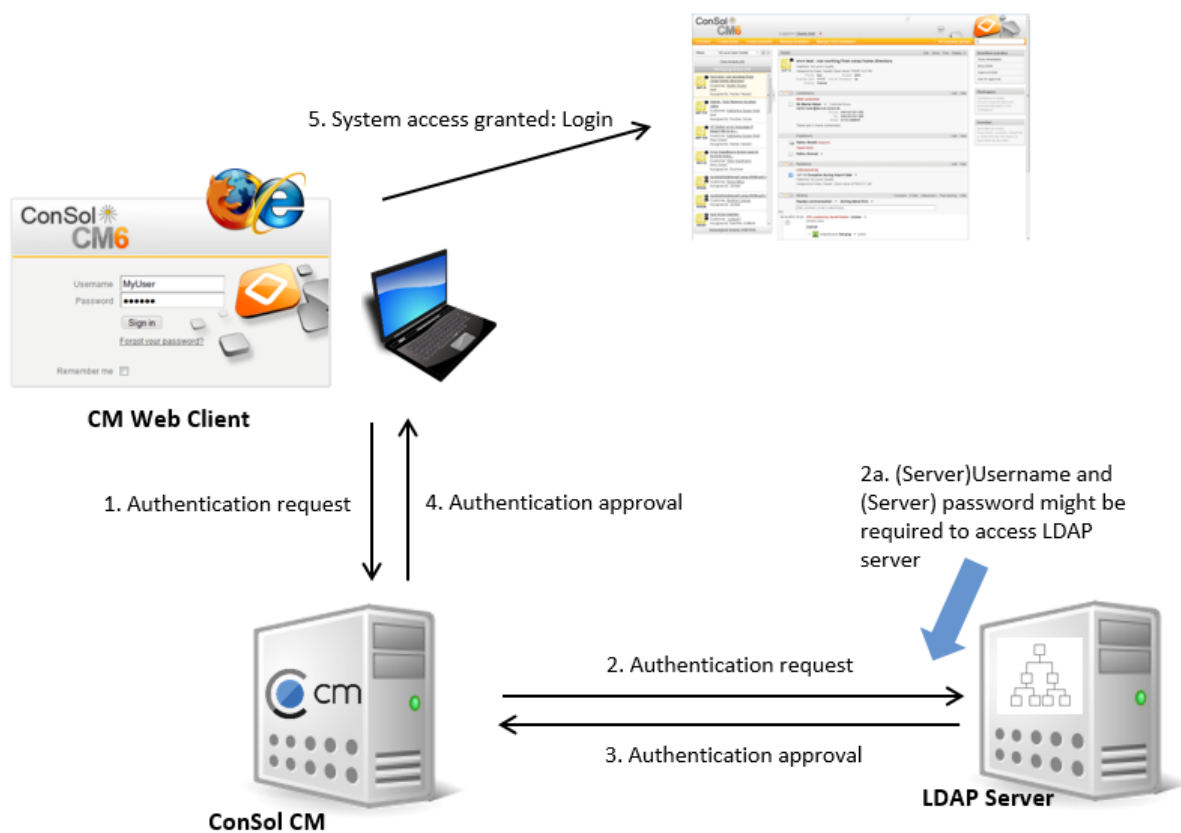


Fig. 3: ConSol CM - LDAP Authentication (Web Client)

## 2.3 Architecture of a CM System with CMRF and DWH

---

- [Overview](#)
- [DWH Database](#)
  - [Oracle](#)
  - [Microsoft SQL](#)
  - [MySQL](#)

### 2.3.1 Overview

In order to allow Business Intelligence (BI) tools or other applications to build specific reports, OLAP cubes, and other analyses, ConSol CM provides a Data Warehouse (DWH) as one of the standard components. The DWH is a separate database (or database schema, see below). The DWH is filled by a Java EE application called *ConSol CM Reporting Framework* (CMRF).

Thus, the ConSol CM standard function set comprises two components which enable reporting:

- **CMRF** (ConSol CM Reporting Framework)  
This is a Java EE application which synchronizes the ConSol CM database with the ConSol CM Data Warehouse (DWH). The CMRF can be deployed into the same application server as the core CM or it can be run on a separate application server. We recommend to work with two application servers whenever CMRF is used. The synchronization of CM data with the DWH can be based on JMS (Java Messaging Service) queues or on direct messaging. For a detailed explanation, please refer to the *ConSol CM Administrator Manual*, section *Using CM for Reporting - Data Warehouse DWH Management*.
- **DWH** (Data Warehouse)  
The ConSol CM DWH is a relational database which can be operated as Oracle, Microsoft SQL Server, or MySQL system (please see *ConSol CM System Requirements* for details). It stores the integrated/pre-processed data from the ConSol CM database. For the configuration of the database connection, please refer to the *ConSol CM Set-Up Manual*, section *Database Preparations for CMRF /DWH Set-Up*.

**Separate** application servers for ConSol CM and CMRF:

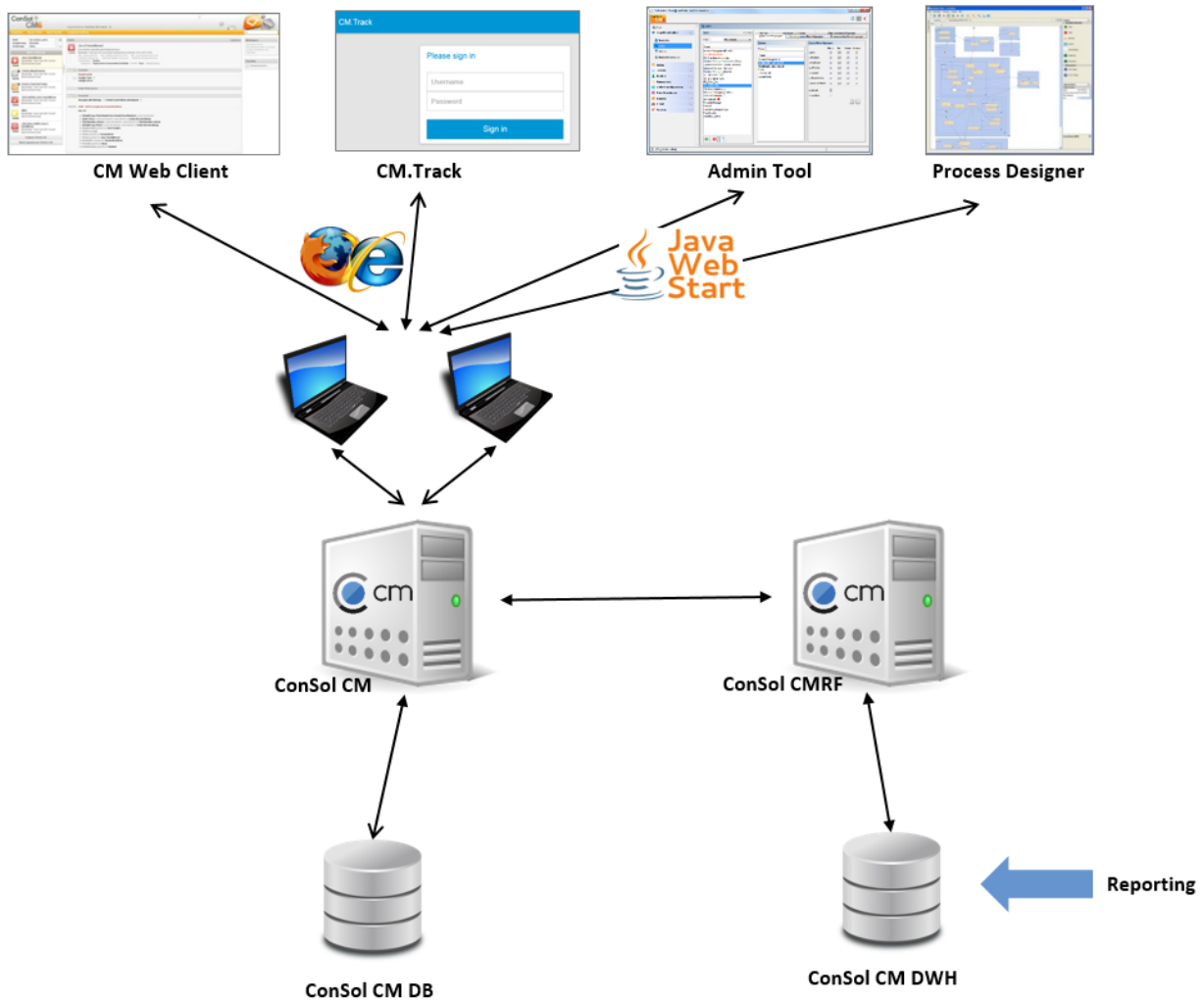


Fig. 1: ConSol CM - Infrastructure with CMRF and DWH (2 Servers)



**One** application server for ConSol CM and CMRF:

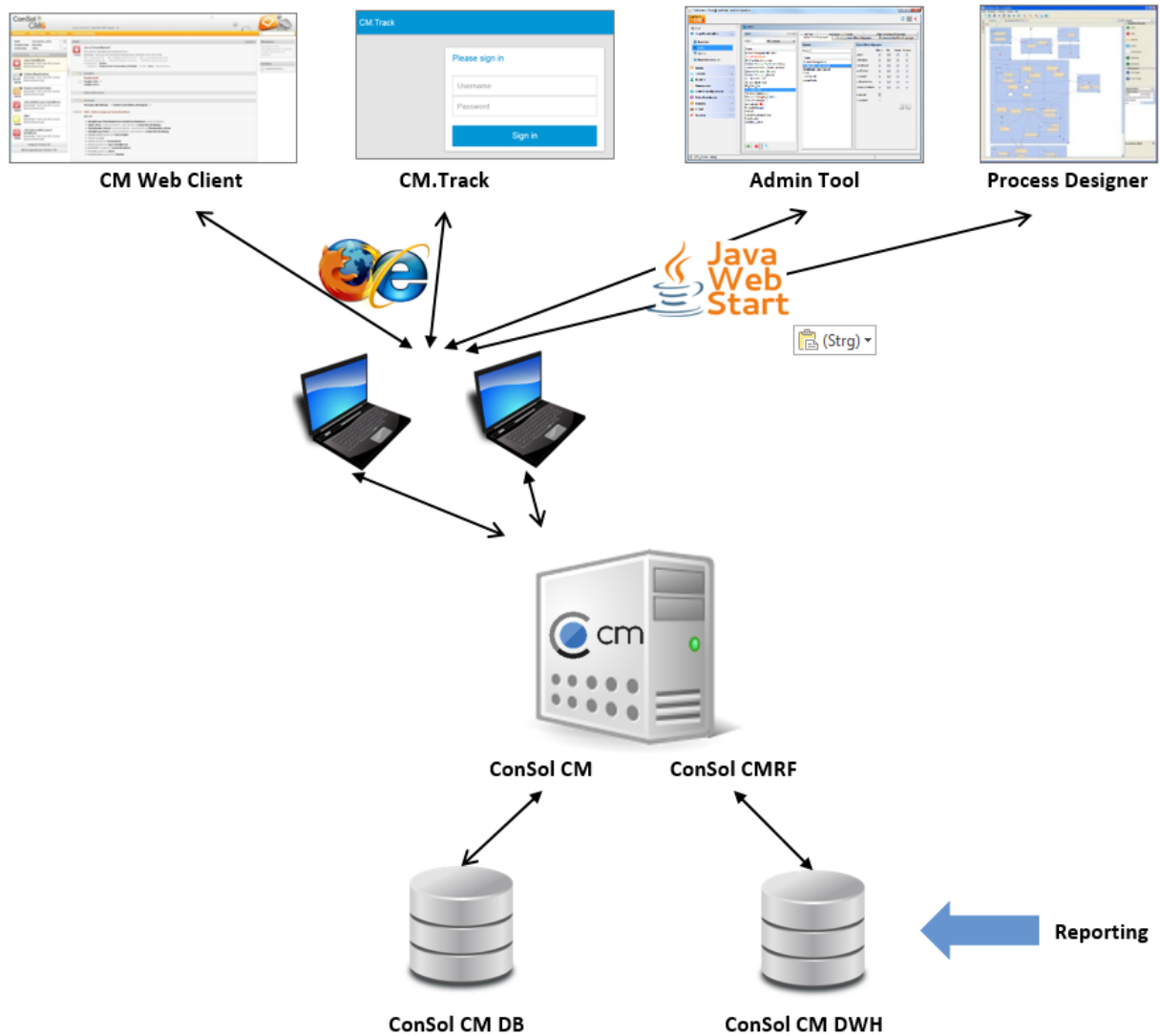


Fig. 2: ConSol CM - Infrastructure with CMRF and DWH (1 Server)

For DWH Management from an operator's point of view, please refer to section [DWH Management](#).

## 2.3.2 DWH Database

### Oracle

One database schema with one database user is used by the DWH.

### Microsoft SQL

One database schema with one database user is used by the DWH.

### MySQL

One database with one database user is used by the DWH.

## 2.4 Architecture of the ConSol CM Application

- [Introduction](#)

### 2.4.1 Introduction

ConSol CM is a Java EE application based on a classical three-tier architecture.

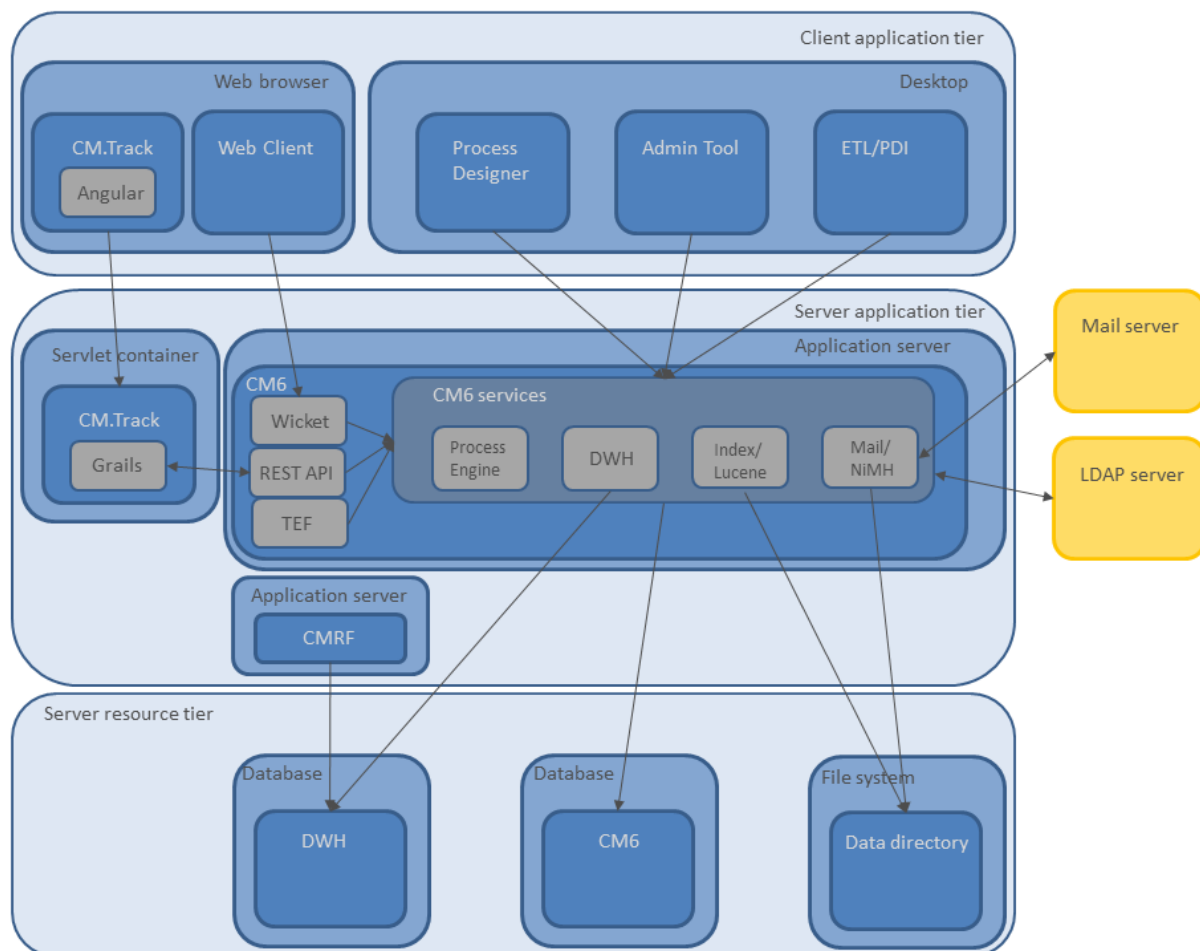


Fig. 1: ConSol CM Application Architecture

## **3 ConSol CM Operations Manual - File System Structure**

---

## 3.1 ConSol CM File System Structure

---

- [Server](#)
  - [ConSol CM Data Directory](#)
  - [JBoss 5 Application Server File Structure](#)
  - [JBoss 7 Application Server File Structure](#)
  - [Oracle WebLogic Application Server File Structure](#)
  - [Log Files](#)
- [Client](#)
- [Variables Used for Standard Path Values in this Manual](#)

### 3.1.1 Server

#### ConSol CM Data Directory

Most of the data concerning the configuration and operation of ConSol CM is stored in the ConSol CM database. However, some data is saved in the file system in the data directory that has been entered during system set-up. The data directory will be called <CMAS\_DATADIR> in this manual.

The following figure and list show examples from a Windows and a Linux system:

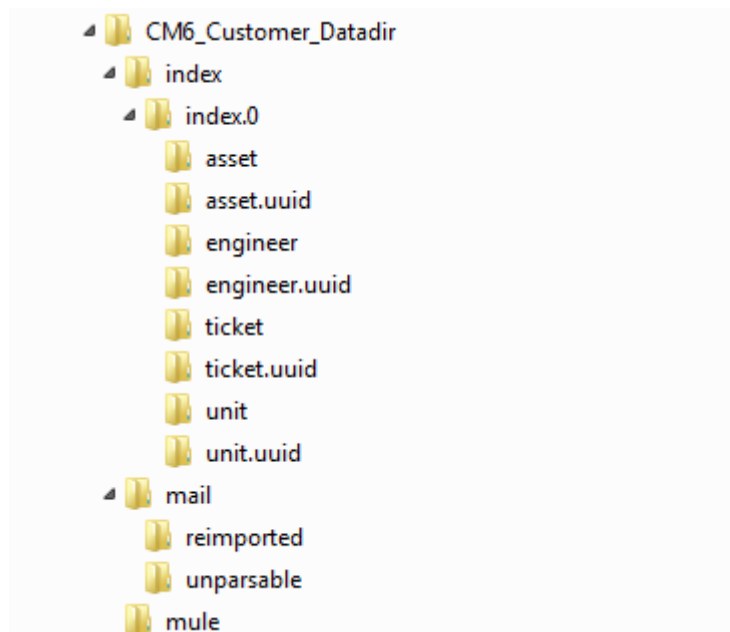


Fig. 1: ConSol CM - Data Directory (Windows)

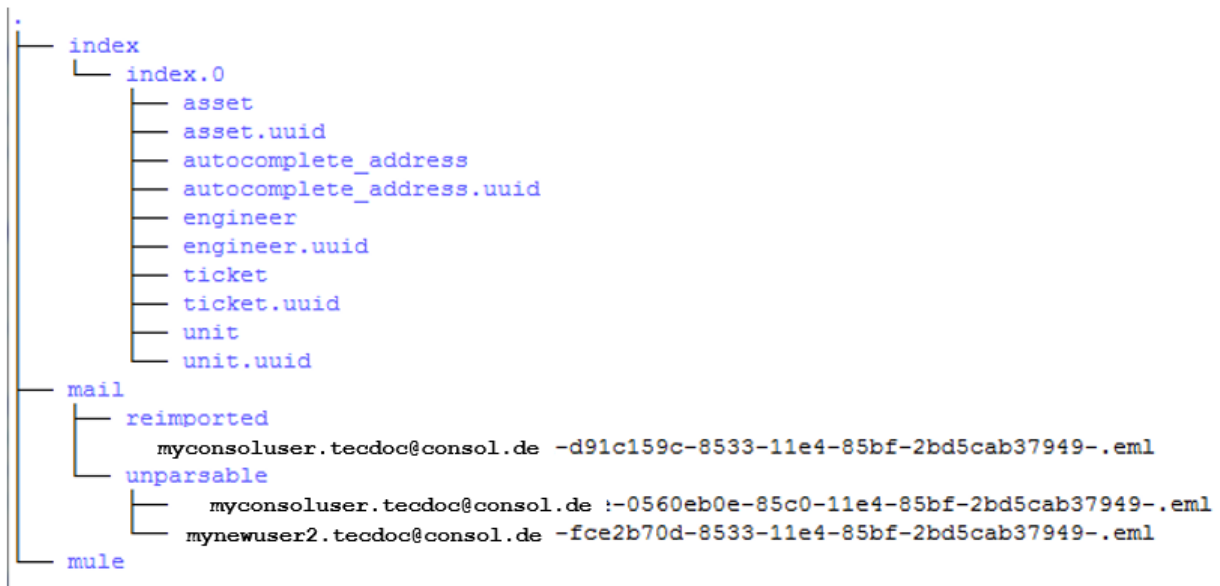


Fig. 2: ConSol CM - Data Directory (Linux)

### Example directories:

- **index**

This is the directory where all the Lucene indexes are stored (see also *ConSol CM Administrator Manual*, section *Search Configuration and Indexer Management*).



Please note that you cannot just include the index directory in your daily CM backup! Rather you have to use the indexer-specific back-up (and restore). Please refer to section [Indexer Configuration, Index Backup and Restore](#).

- **index.0**

In this directory, there is a subdirectory for each required index.

- **mail**

This directory is only relevant when the system is operated in Mule/ESB mode. In case NIMH is used, all data is stored in the database. In this directory, files that are relevant for incoming e-mails are stored.

- **reimported**


In this directory, e-mails are stored that had been stored in the *unparsable* directory and could then be re-imported by a manual action of the administrator.

- **unparsable**

In this directory, incoming e-mails that cannot be processed by the system are stored. They are listed under *E-Mail Backups* in the Admin Tool, see *ConSol CM Administrator Manual*, section *E-Mail Backups*.

- **mule**

This is a directory which might be used for *Mule* (internal *ESB*) data.

 In case units (customer objects) could not be imported during a scene import, there might be *.tmp* files in the data directory.

Example:

```
-rw-rw-r-- 1 hudson hudson 9 Aug 12 16:12 skippedData8260413663331457480.tmp
-rw-rw-r-- 1 hudson hudson 9 Aug 13 09:34 skippedData3491415283536838584.tmp
-rw-rw-r-- 1 hudson hudson 9 Aug 13 10:30 skippedData1311264567017858763.tmp
-rw-rw-r-- 1 hudson hudson 9 Aug 13 11:24 skippedData6517748760975702024.tmp
-rw-rw-r-- 1 hudson hudson 9 Aug 13 13:48 skippedData532044759835774270.tmp
-rw-rw-r-- 1 hudson hudson 9 Aug 13 15:05 skippedData4827028694685189877.tmp
-rw-rw-r-- 1 hudson hudson 9 Aug 17 10:00 skippedData6490396948302181442.tmp
-rw-rw-r-- 1 hudson hudson 9 Aug 25 15:09 skippedData7410776105560129337.tmp
-rw-rw-r-- 1 hudson hudson 9 Sep 16 11:13 skippedData4107847395801839516.tmp
-rw-rw-r-- 1 hudson hudson 9 Sep 22 17:35 skippedData7267475677290414234.tmp
-rw-rw-r-- 1 hudson hudson 9 Sep 23 11:34 skippedData1642119431474300752.tmp
-rw-rw-r-- 1 hudson hudson 9 Sep 23 11:38 skippedData1453186296325724377.tmp
```

## JBoss 5 Application Server File Structure

The following directories are available in a JBoss 5 installation of ConSol CM:

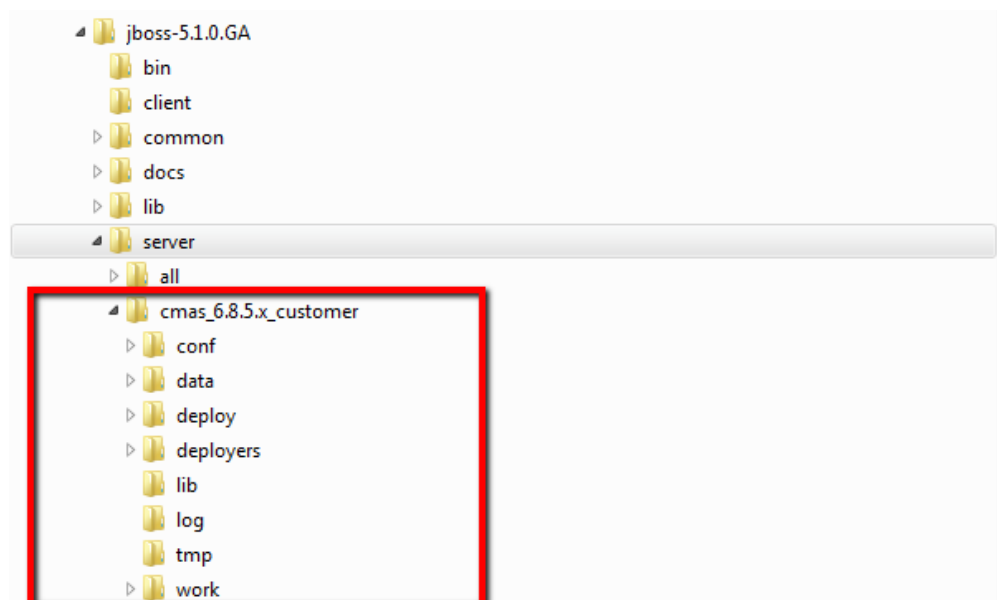


Fig. 3: ConSol CM - File Structure in a JBoss 5 Application Server

**Example directories:**

- **conf**  
Configuration data, e.g.:
  - **jboss-log4j**  
Log file configuration
- **data**  
Data for operation, e.g., *tx-operation* keys
- **deploy**  
Deployed data and configuration data:
  - **cm6.ear**  
Core application, *.ear* file
  - **cm-track.war**  
Application file for the portal ConSol CM.Track
  - **cmDb-ds**  
Database connection configuration
- **deployers**  
Additional deployed application data
- **lib**  
Application-specific libraries, e.g.:
  - **mysql connector**  
In case you use MySQL as a database system.
- **log**  
Log files, see section [Log Files](#).
- **tmp**  
Temporary data
- **work**  
Work directory with a working copy of the application server files. Can be emptied, e.g., for error analysis and/or fixing.

## JBoss 7 Application Server File Structure

The following directories are available in a JBoss 7 installation of ConSol CM:



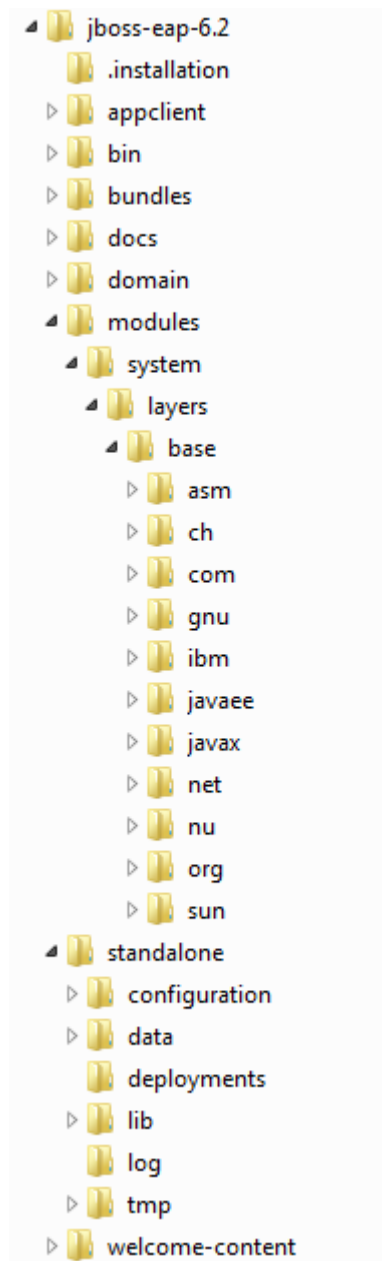


Fig. 4: ConSol CM - File Structure in a JBoss 7 System

**Example directories:**

- **modules\system\layers\base**

Subfolders contain the *JDBC* drivers:

- com\microsoft\sqlserver\jdbc\main\sqljdbc4.jar  
(Microsoft SQL)
- oracle\jdbc\main\ojdbc6-11.2.0.3.jar  
(Oracle)
- com\mysql\jdbc\main\  
(MySQL JDBC driver destination, must be installed manually)

- **standalone**

Configuration in single-server environments:

- **configuration**  
Configuration of the DB connection and logging in the file *cm6.xml*
- **data**  
Data for operation, e.g., *tx-operation* keys
- **deployments**  
Deployed applications, for example, *cm6.ear* and *cm-track.war*
- **log**  
Log files, see section [Log Files](#).
- **tmp**  
Temporary data and also working copy of the application server files. Can be emptied, e.g., for error analysis and/or fixing.

- **domain**

Configuration in domain environments

- **configuration**  
Configuration of the DB connection and logging in the file *domain.xml*
- **servers/<server-name>/log**  
Log files

## Oracle WebLogic Application Server File Structure

In an Oracle WebLogic environment, ConSol CM is installed as a separate domain. ConSol CM as well as CMRF are *managed servers*. Please see the *ConSol CM Set-Up Manual* for details about this configuration.

In the current section, only some directories are explained.

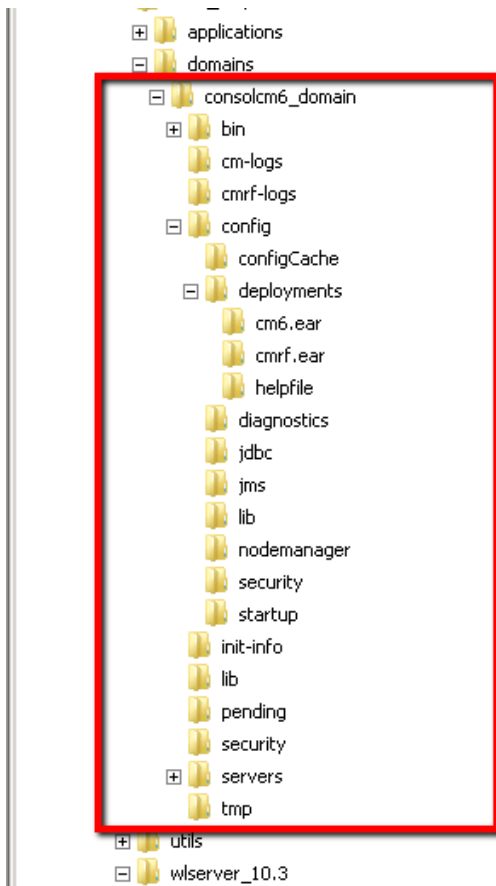


Fig. 5: ConSol CM - File Structure in an Oracle WebLogic Application Server

#### Example directories:

- **bin**  
Start/stop scripts
- **cm-logs**  
All log files except for *cmrf.log*. See also section [Log Files](#).
- **cmrf-logs**
  - **cmrf.log file**  
Log messages for the CMRF (ConSol CM Reporting Framework)
- **config**  
Configuration files
- **deployments**  
Deployed applications, i.e., here: ConSol CM and CMRF as directories

## Log Files

Please refer to the section [ConSol CM Logging and Log Files](#).

### 3.1.2 Client

- **<ENGINEER\_HOME\_DIR>\.cmas\wfeditorR1\var\log**  
The log output of the Process Designer is written to this file. This is only relevant for engineers who work with the Process Designer.
- **<JAVA\_CLIENT\_DIR>**  
The directory where the temporary Java cache is stored. This is used, e.g., for the .jnlp downloads of the Admin Tool and the Process Designer application.

### 3.1.3 Variables Used for Standard Path Values in this Manual

In the current manual, you will often find path names. The following variables are used:

- **<JBoss\_HOME>**  
Defines the home directory of the JBoss application server. Set as environment variable of the operating system. Depending on the context, this might refer to JBoss 5 or JBoss 7.
- **<WEBLOGIC\_HOME>**  
Refers to the home directory of the Weblogic application server.
- **<CMAS\_DATADIR>**  
Refers to the data directory of ConSol CM which is defined during system set-up. This is not an environment variable of the operation system but a CM System property (*cmas-core-shared,data.directory*).
- **<ENGINEER\_HOME\_DIR>**  
Refers to the home directory of the engineer. On windows systems, this is often to be found under C:\users\<USERNAME>, on Linux systems in /home/<USERNAME>.

## 3.2 ConSol CM Logging and Log Files

---

- [Introduction](#)
- [Log Files](#)
  - [Location of the Log Files](#)
  - [Log File Types](#)
  - [Log File Structure](#)
- [Using Log Levels for Troubleshooting](#)
- [Using Logging in Scripts](#)
  - [Logging INFO Messages](#)
  - [Logging DEBUG Messages](#)
- [Configuring the Log Files Using Log4j \(JBoss 5, Weblogic\)](#)
  - [Structure of Log4j XML File \(Example JBoss\)](#)
    - [Appender Section](#)
    - [ConSol CM6-Specific Section <!-- CM -->](#)
- [Configuring the Log Files in JBoss 7 \(EAP 6.x\)](#)

### 3.2.1 Introduction

Log files are the main information source for the administrator about the activities of the system and potential problems of the system. The administrator should have a look at the log files on a regular basis. There may be problems that do not appear on the user interface, but are reported in the log file. Log files can also be an important component in [System Monitoring](#).

### 3.2.2 Log Files

#### Location of the Log Files

The location of the log files can be configured in the respective .xml files:

- **log4j.xml**  
In JBoss 5, where *Log4J* is used as logging framework.
- **cm6.xml** and/or **cm6-cmrf.xml**  
In JBoss 7 standalone, where the built-in logging module of JBoss 7 is used.
- **domain.xml**, tag <subsystem xmlns="urn:jboss:domain:logging:1.3">  
In JBoss 7 in domain mode, where the built-in logging module of JBoss 7 is used.

**Remark for Microsoft Windows systems:**

On a Microsoft Windows system the application server holds a file system lock on the actual log file. In consequence it is not possible to open the file with the usual on board editor *wordpad.exe*, which itself wants to place a lock. You have to create a copy to open it in *WordPad*. We recommend to use a third party editor which does not place locks on the edited files.

## Log File Types

The following log files are used:

- **boot.log**  
Messages concerning system start-up (e.g., the Java version is indicated). JBoss 5 only, application server specific.
- **cmrf.log**  
Messages concerning CMRF (ConSol CM Reporting Framework), i.e., messages that concern the data transfer operations from the ConSol CM database to the CMRF database (DWH). This is done using *JMS* (Java Messaging Service).
- **cmweb.log**  
Messages concerning the ConSol CM Web Client.
- **ctx.log**  
Contains messages of the *Spring Framework*.
- **errors.log**  
Contains only messages that have at least the log level *ERROR*.
- **esb.log**  
Contains messages of the *Mule Framework* (Mule is the internal ESB that is used for the processing of incoming e-mails).
- **ex-reporter.log**  
Messages concerning Hibernate JDBC exceptions are written to this file. The standard configuration (CM 6.10) includes a rotating logger with a maximum number of six copies.
- **index.log**  
Messages concerning the Indexer.
- **mail.log**  
Contains messages of the e-mail subsystem.
- **operationtimes.log**  
Only used when it has been enabled. Contains times of requests in order to identify possible performance bottlenecks.
- **server.log**  
The general log file that contains all messages, as default setting at least with log level *INFO*. It is recommended to use the *DailyRollingFileAppender* in order to prevent the file system from filling up.
- **session.log**  
Contains messages about logins (session starts) and session timeouts of ConSol CM users.

- **sql.log**  
Contains log entries about SQL statements coming from hibernate if it is set to *DEBUG* level (by default it is set to *INFO*).
- **support\_libs\_errors.log**  
Contains errors which are thrown by support libs but are properly handled by the CM application (this method keeps the *server.log* clean).
- **timer-manager.log**  
Contains additional log messages written in log level *DEBUG* when workflow timers are activated or deactivated. Information about the escalation date is logged, too.
- **tx.log**  
Contains *Spring Framework* transactions related log messages.
- **workflow.log**  
Information about activated/reinitialized/deactivated timers is logged with level *INFO* and all debug output related to the workflow engine is written to this dedicated file.

## Log File Structure

In the default configuration, log file entries have the following syntax:

```
Date Timestamp Loglevel [Logger] Message
```

Example for a log file entry (successful start of ConSol CM in JBoss):

```
2012-11-06 14:22:12,685 INFO [e.coyote.http11.Http11Protocol] Starting Coyote HTTP/1.1 on http-0.0.0.0-8080
```

The components of the message:

- **Date:**  
November 6th, 2012
- **Timestamp:**  
14:22:12
- **Loglevel:**  
INFO
- **Logger:**  
e.coyote.http11.Http11Protocol  
Name of a Java class, not complete (only last 30 characters), the real name would be *org.apache.coyote.http11.Http11Protocol*.
- **Message:**  
Starting Coyote HTTP/1.1 on http-0.0.0.0-8080

Simple messages and messages that concern a successful operation often comprise only one line.

When errors occur (log level *ERROR*), you might find stack traces. Please approach one of our ConSol CM consultants or our ConSol CM support team for help.

### 3.2.3 Using Log Levels for Troubleshooting

Sometimes, the pre-defined error levels of the entries in a log file do not produce helpful output, i.e., the information you receive from the log file is not sufficient to help you find the error cause. In this case, you can modify the error level of one or more loggers, e.g., to *DEBUG*.

Please note that in scripts the debug mode has to be switched on explicitly, i.e., the correct code is required, see section [Logging DEBUG Messages](#).

Do not forget to turn the error level back to *INFO* after the error analysis, otherwise your log file(s) might get too big and might fill-up the server hard disk.



#### Information:

##### Log4j:

Note, that changing the log level does not require a server restart. The change in the *log4j* configuration only takes some minutes to get active.

### 3.2.4 Using Logging in Scripts

#### Logging INFO Messages

```
log.info("This is a log output")
```

Writes a message in the following format to the *server.log* file:

```
<Date> <Time> INFO [workflow-element[6][16]-script] This is a logoutput
```

In Groovy it is also allowed to omit the brackets:



```
log.info ">> Ticketname: " + ticket.name
```

## Logging DEBUG Messages

*DEBUG* messages will only appear in the log file when the *log4j* configuration is changed. When logging *DEBUG* messages, make sure to check whether debugging is switched on, to avoid unnecessary operations:

```
if (log.isDebugEnabled()) {  
    log.debug("Processing ticket ${ticket.name}")  
}
```

### 3.2.5 Configuring the Log Files Using Log4j (JBoss 5, Weblogic)

The logging functionality in ConSol CM (for JBoss 5 and WebLogic) is based on *log4j*, a logging framework which is run as a project by the Apache foundation. For detailed information, please refer to the original website under [log4j](#).

The main configuration file of *log4j* is an XML-based text file which is stored under the following path in the application server directory:

- **In JBoss servers:**  
<CMSEVER\_HOME>\conf\jboss-log4j.xml
- **In Oracle WebLogic:**  
<DOMAIN\_HOME>\domains\consolcm6\_domain\log4j.xml

### Structure of Log4j XML File (Example JBoss)

#### Appender Section

In the first section, appenders are defined, i.e., the handles which are used for writing the log files, e.g.:

```

<!-- File appender for the server log -->
<appender name="FILE" class="org.jboss.logging.appender.RollingFileAppender">
  <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
  <param name="File" value="${jboss.server.log.dir}/server.log"/>
  <param name="Append" value="true"/>
  <param name="Threshold" value="INFO"/>
  <param name="MaxFileSize" value="30MB"/>
  <param name="MaxBackupIndex" value="6"/>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{ISO8601} %-5.5p [%-30.30c] [%X{username}-%X
{sessionId}] %m\n"/>
  </layout>
</appender>
or
<appender name="FILE" class="org.apache.log4j.DailyRollingFileAppender">
  <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
  <param name="File" value="${jboss.server.log.dir}/server.log"/>
  <param name="datePattern" value="'_dd-MM-yyyy'.log'"/>
  <param name="Append" value="true"/>
  <param name="Threshold" value="INFO"/>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{ISO8601} %-5.5p [%-30.30c] [%X{username}-%X
{sessionId}] %m\n"/>
  </layout>
</appender>

```

Some parameters are very important here:

- **appender name="FILE"**  
The name of the appender is the reference which is used in other sections of the configuration file.
- **class="org.jboss.logging.appender.RollingFileAppender"**  
The class defines the behavior of the appender. A rolling file appender includes the implementation of a simple round-robin mechanism for log files, so that the unlimited growth of the server log is prevented. An alternative value might be *org.apache.log4j.DailyRollingFileAppender*, which backups automatically the file at the end of the day and starts a new file for the next day.
- **param name="File" value="\${jboss.server.log.dir}/server.log"**  
The location and name of the log file.
- **param name="datePattern" value="'\_dd-MM-yyyy'.log"**  
Determines the pattern, that is used to name the daily backup files (required only for *DailyRollingFileAppender*).
- **param name="Threshold" value="INFO"**  
This is the log level, possible levels are *TRACE*, *DEBUG*, *INFO*, *WARN*, *ERROR*, *ALL*, or *OFF*.
- **param name="MaxFileSize" value="30MB"**  
The maximum file size for this log file. When this limit has been reached, a new log file is started (required only for *RollingFileAppender*).

- **param name="MaxBackupIndex" value="6"**

The maximum number of log files of this type that is kept during the round-robin process. In the example, the 7th log file would be the oldest and would be deleted (required only for *RollingFileAppender*).

## ConSol CM6-Specific Section <!-- CM -->

In the following, CM6-specific log parameters are defined. The first part of the section consists of *<logger!>* paragraphs. Simplified we may say that the *logger name* defines the input channel and the *appender-ref* defines the output channel of the defined log entries.

Some definitions contain a name which indicates a node in the Java class hierarchy. In this case, the definitions (often: the log level) are valid for all log definitions of sub-classes/packages in the class hierarchy and can be overwritten by more specific entries for one or more sub levels.

- **logger name="org.hibernate"**

The following definitions are valid for all sub-classes and packages of *org.hibernate*:

```
<logger name="org.hibernate">
  <level value="INFO"/>
</logger>
```

- **logger name="org.hibernate.util"**

For the sub-class/package *org.hibernate.util* the parameter of *org.hibernate* is overwritten.

```
<logger name="org.hibernate.util">
  <level value="ERROR"/>
</logger>
```

## Another example:

```
<!-- CM/Web -->
<logger name="com.consol.cmweb">
  <level value="INFO"/>
  <appender-ref ref="CMWEB_FILE"/>
  <appender-ref ref="ERROR_FILE"/>
</logger>
```

- **<!-- CM/Web -->**

The explanation of the section.

- **logger name="com.consol.cmweb"**

The name of the Java package(s) for which this logger is defined.

- **appender-ref ref="CMWEB\_FILE"**

An appender to which the log messages should be written. The appender has been defined in the previous section.

- **appender-ref ref="ERROR\_FILE"**

Another appender to which the log messages should be written. The appender has been defined in the previous section.

In the following sections of the *log4j* configuration file, highly specific ConSol CM6 logging parameters are defined. If you would like to work with this parameters, please ask for help of a ConSol CM consultant.

### 3.2.6 Configuring the Log Files in JBoss 7 (EAP 6.x)

In JBoss 7, the configuration of the logging subsystem is done in one of the following files, depending on the infrastructure of your CM system:

- **ConSol CM only:**

<JBoss\_HOME>standalone/configuration/cm6.xml

- **ConSol CM with CMRF/DWH:**

<JBoss\_HOME>standalone/configuration/cm6-cmrf.xml

Usually, you do not have to change any log settings - CM comes with a pre-configured logging subsystem. However, if you really want to change settings: a good introduction into the JBoss EAP Logging System is provided on the Red Hat page about JBoss EAP, [The Logging Subsystem](#).

For configuring the logging behavior, the following section of the config file is relevant:

```
<subsystem xmlns="urn:jboss:domain:logging:1.3">
```

Within the section of this subsystem, the file handlers are defined, each in a separate subsection. Size-rotating file handlers are used with six copies per default.

Here are two examples (*server.log* and *cmweb.log*):

**File handlers for logging subsystem**

```

<size-rotating-file-handler name="FILE" autoflush="true">
  <file relative-to="jboss.server.log.dir" path="server.log"/>
  <append value="true"/>
  <level name="INFO"/>
  <rotate-size value="300m"/>
  <max-backup-index value="6"/>
  <formatter>
    <pattern-formatter pattern="%d %-5.5p [%30.-30c] [%X{username}-%X{sessionId}] %m%n"/>
  </formatter>
</size-rotating-file-handler>

<size-rotating-file-handler name="CMWEB_FILE" autoflush="true">
  <file relative-to="jboss.server.log.dir" path="cmweb.log"/>
  <append value="true"/>
  <rotate-size value="300m"/>
  <max-backup-index value="6"/>
  <formatter>
    <pattern-formatter pattern="%d %-5.5p [%30.-30c] [%X{username}-%X{sessionId}] %m%n"/>
  </formatter>
</size-rotating-file-handler>

```

If you ...

- want to change the size of the single log files:  
Change the value (in MB) of the parameter *rotate-size value*.
- want to change the pattern for the log file entries:  
Change the *pattern-formatter pattern*.
- want to change the number of copies (versions) which are saved:  
Change the value of the parameter *max-backup-index value*.

In the logger section, the target file and log level for each logger are configured, for example:

```

<logger category="com.consol.cmrf">
  <level name="INFO"/>
  <handlers>
    <handler name="CMRF_FILE"/>
    <handler name="ERROR_FILE"/>
  </handlers>
</logger>

```

This means, the logger for classes belonging to *com.consol.cmrf* writes into two files: *CMRF-FILE* and *ERROR\_FILE*. Both have been defined in the subsystem, file handler section above.

If you ...

- want to debug a certain module:  
Change the log level from INFO, e.g., to DEBUG for debugging. Do not forget to set it back to INFO afterwards to avoid writing too many log lines in standard operation mode.

## 4 System Startup and Shutdown of the CM Application Server

---

- [Introduction](#)
- [JBoss 5](#)
  - [Windows](#)
    - [Manual JBoss Startup for CM6 \(Windows 64 Bit\)](#)
    - [Manual Shutdown JBoss for CM6 \(Windows 64 Bit\)](#)
  - [Ubuntu Linux](#)
    - [Manual JBoss Startup for CM6 \(Ubuntu Linux 64 Bit\)](#)
    - [Manual Shutdown JBoss for CM6 \(Ubuntu Linux 64 Bit\)](#)
    - [Manual \(Re-\)Start/Stop Using init Scripts](#)
    - [Manual \(Re-\)Start/Stop Using systemd](#)
- [JBoss 7](#)
  - [Windows](#)
    - [Manual JBoss Startup for CM6 \(Windows 64 Bit\)](#)
    - [Manual Shutdown of the JBoss Server on Windows 64 Bit](#)
  - [Linux](#)
    - [Manual JBoss Startup for CM6 \(Ubuntu Linux 64 Bit\)](#)
    - [Manual Shutdown of the JBoss Server \(Ubuntu Linux 64 Bit\)](#)
  - [General Config](#)
- [WebLogic 11](#)
  - [Admin Server](#)
    - [Windows](#)
    - [Linux](#)
  - [Node Manager](#)
    - [Windows](#)
    - [Linux](#)
  - [Start Managed CM6 Server via WebLogic Administration Console](#)

## 4.1 Introduction

---

ConSol CM can be run on Linux and on Windows systems and is implemented for JBoss and Oracle WebLogic application servers. For a list of supported versions and distributions, please refer to the *System Requirements* of your ConSol CM version.

When you deal with starting/stopping ConSol CM, there are two levels which have to be taken into consideration:

1. The basic start/stop (command line) commands which are - in the end - adapted start/stop commands of the respective application server.
2. The integration of these commands into an environment which allows automatic system start/stop, e. g., writing an */etc/init.d* script for a Linux system or configuring CM as a service on a Windows system.

The start and stop scripts are documented in detail in the *ConSol CM Set-Up Manual*. Here in the current manual, we assume, that you have a CM system which is up and running and that the application server has to be (re-)started/stopped for some reason. Information is provided about the simple command line parameters which can be used to start/stop the system and about how to work with the start scripts.

For the integration of the commands into start scripts or services, i.e., for detailed coding and scripting examples, please refer to the *ConSol CM Set-Up Manual*.



In case there is a severe database problem and the RDBMS has to be restarted, stop the CM system first, then wait until the database system is available again, and then start the CM application server.

The following list of start/stop commands and tips is ordered by flavor of the application server:

- [JBoss 5](#)
- [JBoss 7](#)
- [Oracle WebLogic 11g](#)



## 4.2 JBoss 5

---

### 4.2.1 Windows

#### Manual JBoss Startup for CM6 (Windows 64 Bit)

Run the following command to start the JBoss server (or put it into a *.bat* file):

```
<JBOSS_HOME>\bin\run.bat -c <profile name>
```

**Example:**

```
<JBOSS_HOME>\bin\run.bat -c cmas
```

#### Manual Shutdown JBoss for CM6 (Windows 64 Bit)

Run the following command to stop the JBoss server (or put it into a *.bat* file):

```
<JBOSS_HOME>\bin\shutdown.bat -S
```

### 4.2.2 Ubuntu Linux

#### Manual JBoss Startup for CM6 (Ubuntu Linux 64 Bit)

Run the following command to start the JBoss server (or put it into a start script):

```
<JBOSS_HOME>/bin/run.sh -c <profile name>
```

**Example:**

```
<JBoss_HOME>/bin/run.sh -c cmas
```

## Manual Shutdown JBoss for CM6 (Ubuntu Linux 64 Bit)

Run the following command to stop the JBoss server (or put it into a stop script):

```
<JBoss_HOME>/bin/shutdown.sh -S
```

## Manual (Re-)Start/Stop Using init Scripts

In a great number of systems, there will be *init* scripts. Then you can just use the script in */etc/init.d* for all operations.

```
/etc/init.d/<CM_INIT_SCRIPT> (start|stop|restart)
```

## Manual (Re-)Start/Stop Using systemd

Start, stop and restart are performed using the systemd-specific file. Please refer to the *ConSol/CM Set-Up Manual* for information about how to configure ConSolCM with systemd.

To start/stop/restart the system, simply use the following command:

```
systemctl (start|stop|restart) <CM_STARTSCRIPT>
```

## 4.3 JBoss 7

---

### 4.3.1 Windows

#### Manual JBoss Startup for CM6 (Windows 64 Bit)

Run the following command to start the JBoss server (or put it into a start script).

The *server-config* file is:

- **cm6.xml**  
in a CM6-only installation (shown in the example)
- **cm6-cmrf.xml**  
in a CM environment with CMRF/DWH

```
<JBOSS_HOME>\bin\standalone.bat --server-config=cm6.xml -b=0.0.0.0
```

#### Manual Shutdown of the JBoss Server on Windows 64 Bit

Run the following command to stop the JBoss server (or put it into a stop script):

```
<JBOSS_HOME>\bin\jboss-cli.bat --connect --command=:shutdown
```

### 4.3.2 Linux

#### Manual JBoss Startup for CM6 (Ubuntu Linux 64 Bit)

Run the following command to start the JBoss server (or put it into a start script).

The *server-config* file is:

- **cm6.xml**  
in a CM6-only installation (shown in the example)
- **cm6-cmrf.xml**  
in a CM environment with CMRF/DWH

```
<JBoss_HOME>\bin\standalone.sh --server-config=cm6.xml -b=0.0.0.0
```

## Manual Shutdown of the JBoss Server (Ubuntu Linux 64 Bit)

Run the following command to stop the JBoss server (or put it into a stop script):

```
<JBoss_HOME>\bin\jboss-cli.bat --connect --command=:shutdown
```

### 4.3.3 General Config

- In case *-b=localhost* or *-b=127.0.0.1* is set as parameter, ConSol CM6 will only be accessible from the same server, where JBoss is running.
- Enter the network IP or network name to make CM6 accessible from outside. In this case, the CM6 URL won't be accessible from inside with localhost-url.
- If you enter *-b=0.0.0.0* the server will be accessible from outside and inside over server-url and localhost-url.

When the server is started you will get the following message in *server.log*:

```
2016-08-15 13:45:34,538 INFO [e.coyote.http11.Http11Protocol] [-] JBWEB003001: Coyote HTTP/1.1
initializing on : http-0.0.0.0:8380
2016-08-15 13:45:34,588 INFO [e.coyote.http11.Http11Protocol] [-] JBWEB003000: Coyote HTTP/1.1
starting on: http-0.0.0.0:8380
2016-08-15 13:45:34,627 INFO [org.jboss.as.remoting] [-] JBAS017100: Listening on 127.0.0.1:102
99
```

## 4.4 WebLogic 11

---

There are three servers which have to be started, before you can access CM6:

- Admin server
- Node manager
- Managed server

### 4.4.1 Admin Server

#### Windows

If you have chosen *production mode* during set-up and you do not want to type the WebLogic admin user and password during admin server startup, do the following:

- Search the domain folder (configured during set-up) and edit the admin server start file under `<wls_home>\user_projects\domains\<domain Name>\startWebLogic.cmd`.
- Add below the line with 'SETLOCAL':

```
set WLS_USER=<wls_adminuser>

set WLS_PW=<wls_adminuser_password>
```

- Replace `<>` with your chosen values during set-up.

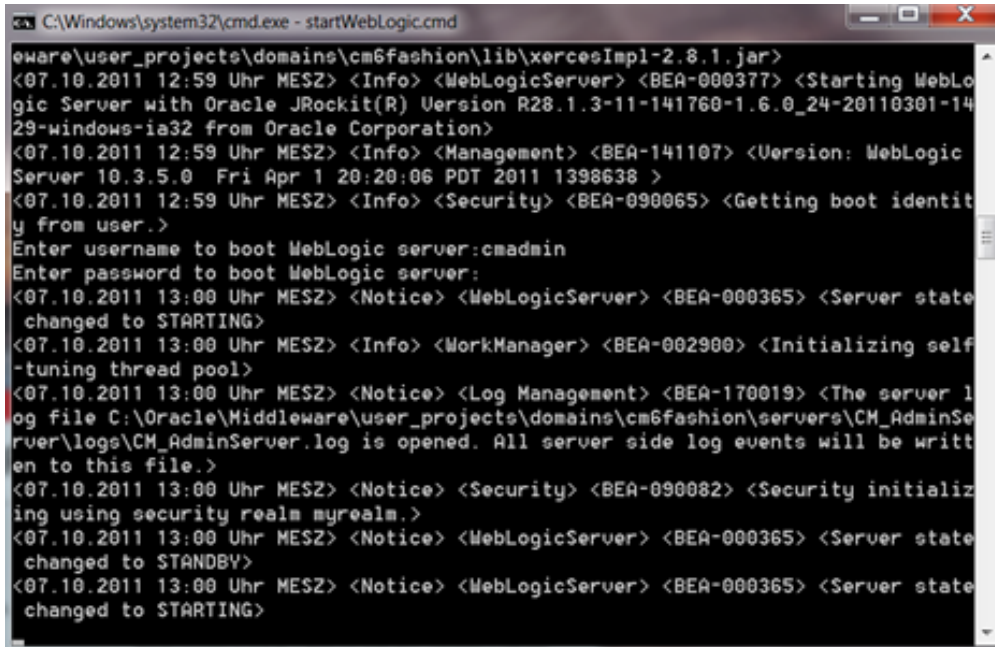


Note that changes in this file can be overwritten by the WebLogic Configuration Wizard. Backup this file before using the Configuration Wizard.

Note that this a password in plain text! This should not be used in production environments!

- Start the admin server with:

```
<WLS_HOME>\user_projects\domains\<domain Name>\startWebLogic.cmd
```



```
C:\Windows\system32\cmd.exe - startWebLogic.cmd
eware\user_projects\domains\cm6fashion\lib\xercesImpl-2.8.1.jar>
<07.10.2011 12:59 Uhr MESZ> <Info> <WebLogicServer> <BEA-000377> <Starting WebLo
gic Server with Oracle JRockit(R) Version R28.1.3-11-141760-1.6.0_24-20110301-14
29-windows-ia32 from Oracle Corporation>
<07.10.2011 12:59 Uhr MESZ> <Info> <Management> <BEA-141107> <Version: WebLogic
Server 10.3.5.0 Fri Apr 1 20:20:06 PDT 2011 1398638 >
<07.10.2011 12:59 Uhr MESZ> <Info> <Security> <BEA-090065> <Getting boot identit
y from user.>
Enter username to boot WebLogic server:cmadmin
Enter password to boot WebLogic server:
<07.10.2011 13:00 Uhr MESZ> <Notice> <WebLogicServer> <BEA-000365> <Server state
changed to STARTING>
<07.10.2011 13:00 Uhr MESZ> <Info> <WorkManager> <BEA-002900> <Initializing self
-tuning thread pool>
<07.10.2011 13:00 Uhr MESZ> <Notice> <Log Management> <BEA-170019> <The server l
og file C:\Oracle\Middleware\user_projects\domains\cm6fashion\servers\CM_AdminSe
rver\logs\CM_AdminServer.log is opened. All server side log events will be writt
en to this file.>
<07.10.2011 13:00 Uhr MESZ> <Notice> <Security> <BEA-090082> <Security initializ
ing using security realm myrealm.>
<07.10.2011 13:00 Uhr MESZ> <Notice> <WebLogicServer> <BEA-000365> <Server state
changed to STANDBY>
<07.10.2011 13:00 Uhr MESZ> <Notice> <WebLogicServer> <BEA-000365> <Server state
changed to STARTING>
```

Fig. 1: Starting CM with Weblogic, 1


## Linux

If you have chosen *production mode* during set-up and you do not want to type the WebLogic admin user and password during admin server startup, do the following:

- Search the domain folder (chosen during set-up) and edit the admin server start file under `<wls_homer>\user_projects\domains\<domain Name>\startWebLogic.sh`.

- Add the two lines beginning with *WLS\_USER* and *WLS\_PW*, depending on the chosen values during domain set-up.

```
#!/bin/sh
# WARNING: This file is created by the Configuration Wizard.
# Any changes to this script may be lost when adding extensions to this configuration.
WLS_USER=<your domain admin user>
WLS_PW=<your domain admin password>
DOMAIN_HOME="../../middleware/user_projects/domains/<your domain>"
${DOMAIN_HOME}/bin/startWebLogic.sh $*
```

 Note that changes in this file can be overwritten by the WebLogic Configuration Wizard. Backup this file before using the Configuration Wizard.

- Go to the directory of *startWeblogic.sh* (*<WLS\_HOME>user\_projects\domains\<domain Name>\startWebLogic.cmd*). Start the admin server with:

```
sh startWebLogic.sh
```

## 4.4.2 Node Manager

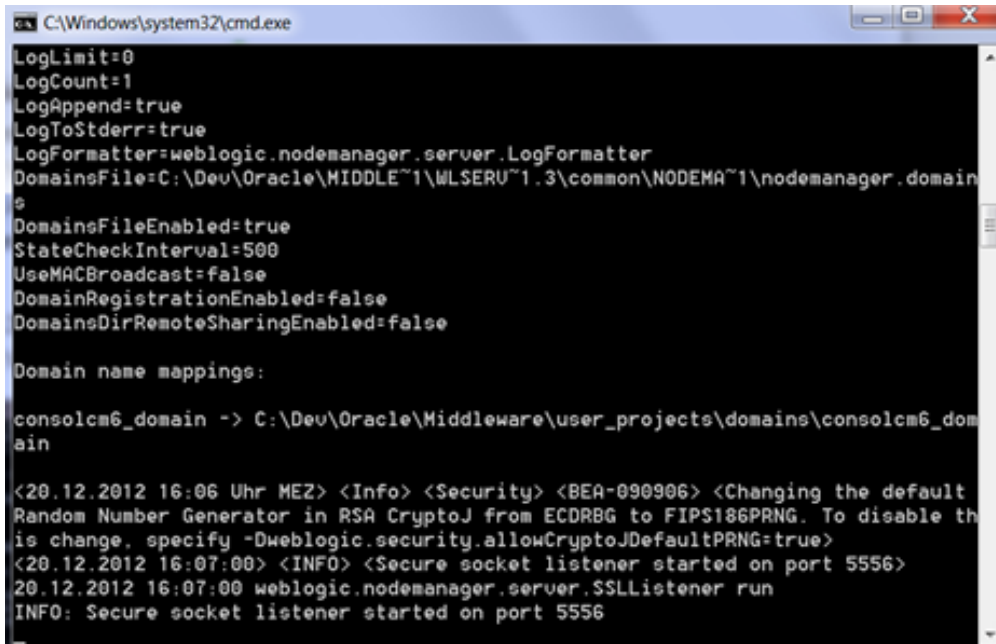
Only if the node manager is running, the start of the managed server via WebLogic Administration Console is possible.

If you want to start the managed server by command line or service, the node manager is optional.

## Windows

- Start the WebLogic node manager with:

```
<WLS_HOME>\wlserver_10.3\server\bin\startNodeManager.cmd
```



```
C:\Windows\system32\cmd.exe
LogLimit=0
LogCount=1
LogAppend=true
LogToStderr=true
LogFormatter=weblogic.nodemanager.server.LogFormatter
DomainsFile=C:\Dev\Oracle\MIDDLE~1\WLSERU~1.3\common\NODEMA~1\nodemanager.domains
DomainsFileEnabled=true
StateCheckInterval=500
UseMACBroadcast=false
DomainRegistrationEnabled=false
DomainsDirRemoteSharingEnabled=false

Domain name mappings:

consolcm6_domain -> C:\Dev\Oracle\Middleware\user_projects\domains\consolcm6_domain

<20.12.2012 16:06 Uhr MEZ> <Info> <Security> <BEA-090906> <Changing the default
Random Number Generator in RSA CryptoJ from ECDRBG to FIPS186PRNG. To disable th
is change, specify -Dweblogic.security.allowCryptoJDefaultPRNG=true>
<20.12.2012 16:07:00> <INFO> <Secure socket listener started on port 5556>
20.12.2012 16:07:00 weblogic.nodemanager.server.SSLListener run
INFO: Secure socket listener started on port 5556
```

Fig. 2: Starting the Node Manager

## Linux

- Go to the following directory:

```
<WLS_HOME>\wlserver_10.3\server\bin\
```

- Run the node manager with the following command:

```
sh startNodeManager.sh
```



### 4.4.3 Start Managed CM6 Server via WebLogic Administration Console

- Log in to the WebLogic Administration Console with the following URL:  
*<http://localhost:7001/console/login/LoginForm.jsp>*

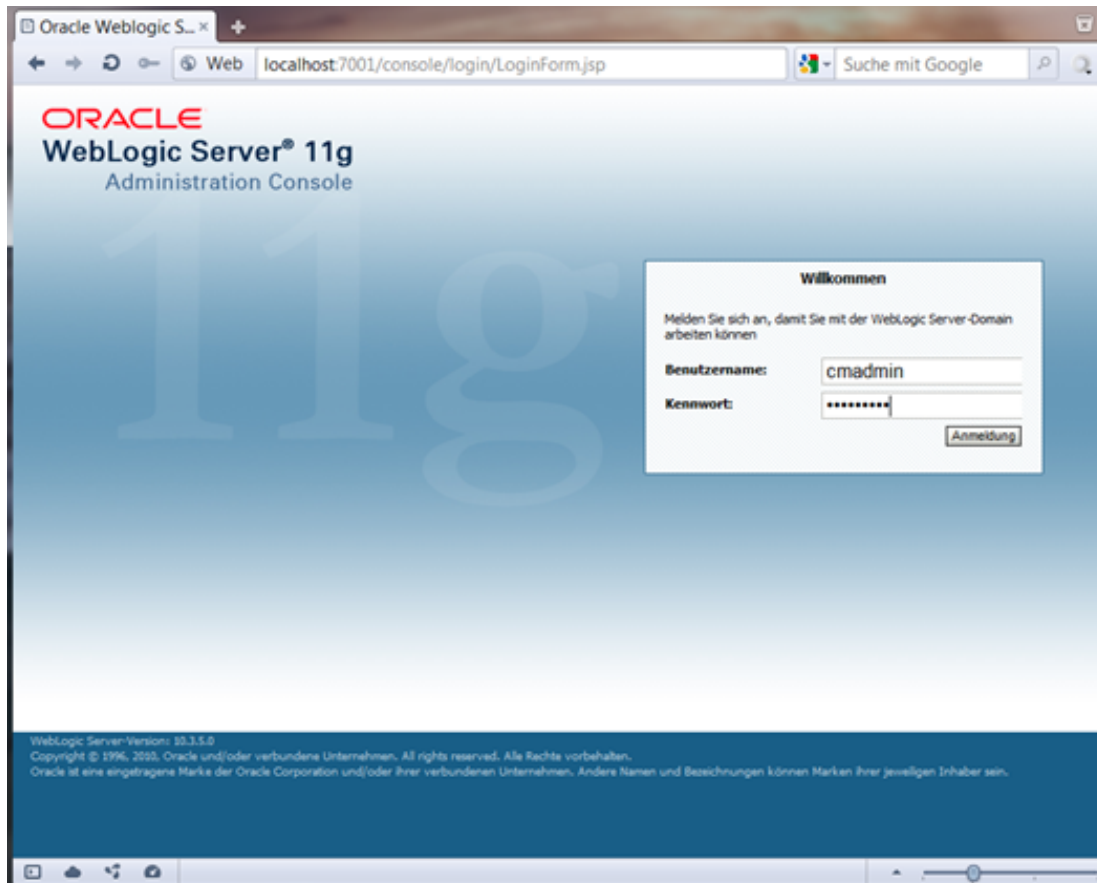


Fig. 3: Starting CM using the WLS Admin Console, 1

- Look at the domain structure and choose *consolcm6\_domain/Environnement/Servers*.
- Choose *Control*

- Tick the checkbox from *CM\_ManagedServer\_1* and start the server.

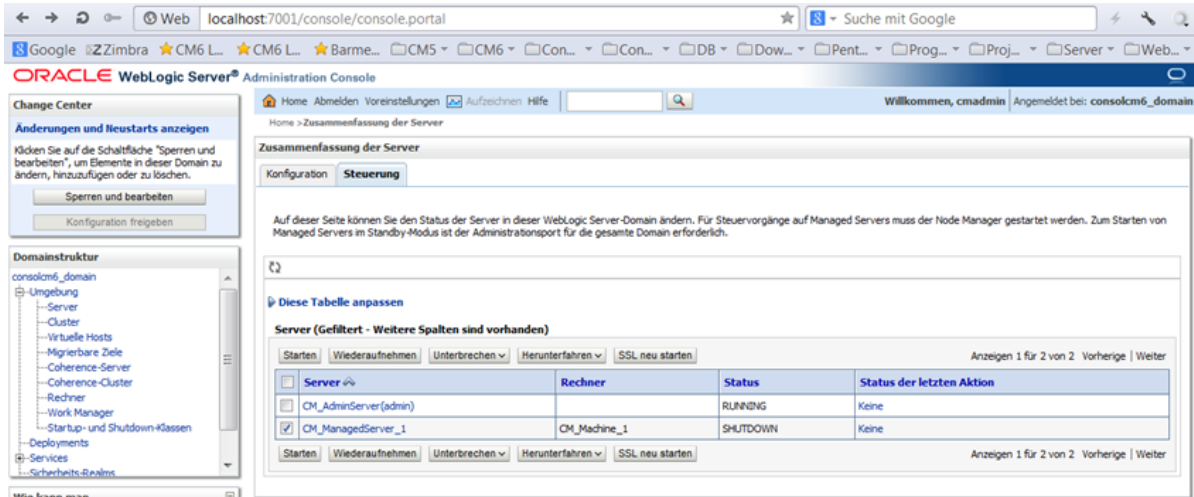


Fig. 4: Starting CM using the WLS Admin Console, 2

- When asked because of server start, confirm with *Yes*.



Fig. 5: Starting CM using the WLS Admin Console, 3

- The server is changing to mode *starting*.

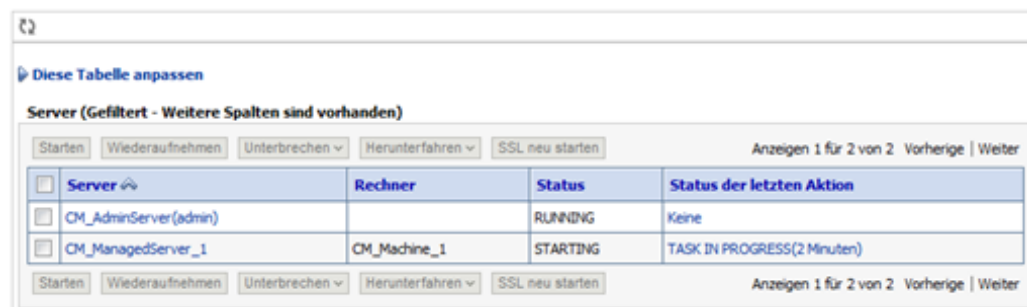


Fig. 6: Starting CM using the WLS Admin Console, 4

- With some configurations the first server start may fail (*Error 404--Not Found* on CM6 start page).
- In this case just shutdown the server *CM\_ManagedServer\_1* and start it again.
- Start the CM6 configuration on the following browser page:  
*http://localhost:7003/*



## 5 Data Backup, Restore, and Recovery Procedures

---

- [Introduction](#)
- [Backup and Restore of the ConSol CM Configuration](#)
- [Index Backup and Restore](#)
- [\(Backup and\) Restore of a Completely Damaged DWH Database](#)

## 5.1 Introduction

---

ConSol CM is a Java EE application with a classic three-tier architecture. Therefore, in general all standard mechanisms for data backup and restore of a Java EE application server with a database connection apply. Some specific information about ConSol CM is explained in this section

## 5.2 Backup and Restore of the ConSol CM Configuration

---

In addition to a database backup on database level, ConSol CM also allows to export and back up a so called **scenario** using the Admin Tool. Depending on the export parameters, this scenario contains either the complete system configuration (Admin Tool configuration and workflows) or only a part of this configuration. It is also possible to include runtime data (tickets, customers, and resources) in the export. Nevertheless, this should not be done when exporting large environments, as this is only intended for transferring small amounts of data, as test tickets and customers. Therefore, a scenario export does not replace a database backup. Usually, a scenario is used to transfer data from a test environment to a staging or production environment. But you can also use it to back up the complete configuration of the (production) system. In case the restore of a database backup fails, you will then have the possibility to quickly restore the configuration. Nevertheless, tickets, customers, and resources will not be available afterwards.

A scenario will not contain the values of CM System Properties which are not in a customer-specific module, e.g., the mail server and LDAP configuration will not be exported. A detailed explanation about which parameters are exported in a scenario is provided in the *ConSol CM Administrator Manual*, chapter *Deployment*.

Therefore, ConSol recommends the following: Export a scenario of the production environment on a regular basis and save it on a share which is included in the backup system. In order to save the values of the CM System Properties, either use a simplified solution and store a screenshot of the Admin Tool values or export the table *cmas\_configuration* of the CM database.

Please also refer to the *ConSol CM Administrator Manual*, chapter *Deployment* for further details and background information about working with scenarios.

## 5.3 Index Backup and Restore

---

This is explained in detail in section [Indexer Configuration, Index Backup and Restore](#).

## 5.4 (Backup and) Restore of a Completely Damaged DWH Database

---

The ConSol CM DWH (Data Warehouse) is filled by the CMRF (Consol CM Reporting Framework) either triggered manually by an administrator (in *ADMIN* mode) or on-the-fly in *LIVE* mode. For a detailed explanation, please refer to the *ConSol CM Administrator Manual*, section *Data Warehouse (DWH) Management*. In case the DWH is corrupt or has to be restored for other reasons, the DWH can be re-built completely using the Admin Tool, DWH Configuration, *Initialize* with the option *overwrite* enabled. This recreates the complete database scheme. Afterwards, use *Transfer* for the initial filling of the DWH. However, this method has a rather great impact on the live operation of the DWH (i.e., a downtime which might extend to some hours or even days) and should only be used if the DWH cannot be updated using the *Update* option.

It depends on the size of the DWH if it makes sense to include the DWH into the regular database backup. If the DWH is large, it should be part of the regular database backup, because a complete recreation would take too long to be compatible with every-day reporting requirements. For a rather small DWH, it might be easier and quicker to restore it using the recreation as explained in the previous paragraph. Since configuration, infrastructure, and performance are highly system-specific, we ask you to talk to your ConSol CM consultant to discuss which is the best solution for your company.



## 6 System Update (Minor Version)

---

- [Introduction: Types of CM Installations](#)
- [Updating a ConSol CM6 Standard Installation \(for Case 1\)](#)
  - [Information about the Update](#)
  - [Getting the Required Files](#)
  - [Perform the Update](#)

## 6.1 Introduction: Types of CM Installations

---

For ConSol CM, there are two types of installations/instances. Depending on which type of installation you operate in your company, you can perform a system update yourself or not.

- **Case 1: Standard**

You have a standard configuration, i.e., the complete customizing has been performed using standard CM tools (Admin Tool configuration, Process Designer for workflows). Neither the *.ear* files for CM6 and CMRF, nor the *.war* file for CM.Track have been modified. In this case, you can perform the update yourself as described in the following sections.

- **Case 2: Custom**

You have a customized ConSol CM system, i.e., files within either the *cm6.ear*, the *cmrf.ear*, and/or files within the *cm-track.war* have been modified and a new *.ear/.war* file has been built. Then a custom-specific development project is managed at ConSol's and you should ask your technical ConSol CM consultant for support.


## 6.2 Updating a ConSol CM6 Standard Installation (for Case 1)

---

The following paragraphs list all the steps you have to perform in case of a CM update.

### 6.2.1 Information about the Update

As a first step, please read all the *Release Notes* which cover the versions between source version (which is currently installed on your system) and target version (which you want to install now). The first sections of each *Release Notes* document deal with the required steps in case of a system update. Follow the instructions provided in these sections.

In some cases, it might be required to adapt the CM system to a new version, e.g., to modify scripts when Groovy methods of the CM API have become obsolete. A hint about these changes will be provided in the *Release Notes*. Please ask your ConSol CM consultant for help and prepare everything which is required before  the update starts!

### 6.2.2 Getting the Required Files

Depending on the components you have installed in your CM system, the following files are required for an update. Please ask the ConSol support team or your ConSol CM consultant to get the files of the new CM version. If your company has a maintenance contract with ConSol, you can download the required files from the ConSol ftp server.

1. For every installation:  
the **CM6 .ear file**:  
`dist-package-ear-<CM version>.ear`, e.g., `dist-package-ear-6.10.5.2.ear`
2. If DWH/CMRF are installed:  
the **CMRF .ear file**  
`cmrf-package-ear-<CM version>.ear`, e.g., `cmrf-package-ear-6.10.5.2.ear`
3. If CM.Track V1 is installed:  
the **new .war file**:  
`cmtrack-<version-number>.war`, e.g., `cmtrack-6.10.5.1.war`
4. If CM.Track V2 is installed:  
the **new .war file** for a web container:  
`cmtrack-v2-distribution-<version-number>.war`, e.g., `cmtrack-v2-distribution-6.10.5.1.war`
5. If ETL is used (e.g., for import of customer or engineer data into the CM database):  
the **CM ETL package**:  
`etl-package-distribution-<version-number>-kettle.zip`, e.g., `etl-package-distribution-6.10.5.1-kettle.zip`

## 6.2.3 Perform the Update



Please note that if you perform an update, you have to perform the version-specific actions (mentioned in the first sections of each *Release Notes* document) for all versions between source and target version.

- Shutdown the ConSol CM system.
- Do everything that is necessary according to all *CM Release Notes* between the source and the target version, in ascending order.
- Deploy the new CM *.ear* and *.war* files.
- (Re-)Start the CM system.

## 7 Management of E-Mail Functionalities

---

- [Introduction](#)
- [Short Overview of CM Components Relevant for Mailing](#)
  - [E-Mail Has to Be Fetched](#)
    - [Mule/ESB Mode](#)
    - [NIMH Mode](#)
  - [E-Mail Has to Be Sent](#)
- [Fine-Tuning CM Mailing](#)
  - [Changing Administrator E-Mail Addresses](#)
  - [Changing Mailing Parameters](#)
    - [General Information about Changing Mailing Parameters](#)
    - [Example 1: Changing the Maximum Size of E-Mail Attachments](#)
    - [Example 2: Narrowing Down the File Types Which Are Allowed as Attachments](#)
- [Monitoring E-Mail Functionalities](#)

## 7.1 Introduction

---

One of the core ConSol CM functionalities is the interaction with one or more mail servers to fetch e-mails and to send e-mails. A short explanation of the ConSol CM system architecture, including the e-mail interactions, is provided in the current manual in section [System Overview](#).

This should be sufficient to get a basic understanding of the topic and to be able to start/stop the relevant components.

However, if you would like to get some deeper knowledge about the topic, you should read the *ConSol CM Administrator Manual*, section *E-Mail Configuration*, maybe even the section *Admin Tool Scripts, Scripts of Type E-Mail*.

## 7.2 Short Overview of CM Components Relevant for Mailing

---

In order to run smoothly, the following modules of CM have to be taken into consideration on a server (the e-mail configuration of a CM cluster is explained in detail in the *ConSol CM Set-Up Manual*).

1. E-mail has to be fetched.
2. E-mail has to be sent.

The E-Mail configuration is explained in detail in the *ConSol CM Administrator Manual*, section *E-Mail*. The CM system properties which are relevant are summarized in the *ConSol CM Administrator Manual*, *Appendix D (Important System Properties - Ordered by Area of Application)*, section *E-Mail Configuration*.

### 7.2.1 E-Mail Has to Be Fetched

First, you have to know which module is used for fetching e-mails, i.e., in which mode CM is run:

- Mule/ESB mode
- NIMH mode

If you are not sure in which mode the CM is running, check the CM System property *cmas-core-server.nimh.enabled*. If this is set to *true*, your system is running in NIMH mode, if it is set to *false*, your system is running in Mule/ESB mode.

#### Mule/ESB Mode

The internal ESB is used for fetching mail.

- **Service:**  
The CM ESB Services are active (see Admin Tool, navigation group *Services*, navigation item *ESB Services*).
- **Scripts:**  
The mailing scripts without the NIMH extension are active (e.g., *CreateTicket.groovy*), see Admin Tool, navigation group *System*, navigation item *Scripts*.
- **Data:**  
E-mails which cannot be processed are stored in the file system under *<CMAS\_DATADIR>/mail/unparsable*.

#### NIMH Mode

- **Service:**  
The NIMH service is active (see Admin Tool, navigation group *Services*, navigation item *CM Services*).

- **Scripts:**

The mailing scripts with the NIMH extension are active (e.g., *NIMHCreateTicket.groovy*), see Admin Tool, navigation group *System*, navigation item *Scripts*.

- **Data:**

E-mails which cannot be processed are stored in the database, table *cmas\_nimh\_archived\_mail*.

## 7.2.2 E-Mail Has to Be Sent

When the CM server is started, the module which can send e-mails using the SMTP server is started as well. Sending out e-mails is completely independent of the Mule/ESB vs. NIMH mode and is always active

- when CM is up and running.
- if a correct value has been set for the SMTP server (CM System Property *cmas-core-server.mail.smtp.email*).



## 7.3 Fine-Tuning CM Mailing

---

### 7.3.1 Changing Administrator E-Mail Addresses

This is explained in detail in the *ConSol CM Administrator Manual, Appendix E - Administrator and Notification E-Mail Addresses*.

### 7.3.2 Changing Mailing Parameters

#### General Information about Changing Mailing Parameters

The basic mailing parameters are changed using the Admin Tool, navigation group *E-Mail*, navigation item *E-Mail*. In case you would like to set some specific values which cannot be reached using the standard graphical Admin Tool interface, you can work with system properties (see Admin Tool, navigation group *System*, navigation item *System Properties*). The CM system properties which are relevant are summarized in the *ConSol CM Administrator Manual, Appendix D (Important System Properties - Ordered by Area of Application)*, section *E-Mail Configuration*. Some properties which are used quite often are mentioned here as examples.

#### Example 1: Changing the Maximum Size of E-Mail Attachments

**System Property:**

Use *cmas-core-serverattachment.max.size*. This sets the maximum attachment size, in MB.

This is a validation property of the CM API. It controls the size of attachments at tickets, at units, and at resources. It also controls the size of incoming ⚠ (not outgoing!) e-mail attachments in NIMH as well as in Mule/ESB mode

#### Example 2: Narrowing Down the File Types Which Are Allowed as Attachments

**System Property:**

Use *cmas-core-server, attachment.allowed.types*. This is a comma-separated list of allowed filename extensions. If no value is defined, all file extensions are allowed.

## 7.4 Monitoring E-Mail Functionalities

---

See section [System Monitoring, Monitoring E-Mail Functionalities](#).

## 8 Indexer Management

---

- [Introduction](#)
- [Indexer Services](#)
- [Indexer Directory Structure](#)
- [Indexer Architecture and Update Principle](#)
  - [Basic Principle](#)
  - [Indexer Update, Step 1: Persistent Store Is Filled](#)
    - [Index Persistent Store as JMS Queue](#)
    - [Index Persistent Store as Database Table](#)
  - [Indexer Update, Step 2: Master Server Update](#)
  - [Administrative Changes](#)
  - [Indexer Update, Step 3: Slave Server Synchronization](#)
  - [Indexer Info about Manual Index Updates \(Using the Admin Tool\)](#)
- [Indexer Restart](#)
- [Index Update Failure Scenarios](#)
  - [Master Failover](#)
    - [Master Failover Active](#)
    - [No Master Failover](#)
  - [Failure of Update Task](#)
- [Monitoring the Indexer](#)
- [Index Backup and Restore](#)
  - [Index Backup](#)
  - [Index Restore](#)
- [System Properties Which Are Relevant for the Indexer Master/Slave Synchronization](#)
- [JBoss Parameters Which Are Relevant for the Indexer Master/Slave Synchronization](#)

## 8.1 Introduction

---

In order to improve the performance of search operations, ConSol CM uses Lucene indices. ConSol CM stores most of its data in a relational database. However, the indices are stored on the file system, in a subdirectory of the data directory (compare section [ConSol CM File System Structure](#), [ConSol CM Data Directory](#)). The ConSol CM module which creates and manages the indices is called the *CM Indexer*.

The following section will provide an overview of the basic principle of the Indexer and will provide some tips and tricks for CM Indexer configuration. The management of the Indexer using the Admin Tool will not be described here, because it is described in great detail in the *ConSol CM Administrator Manual*. Please refer to section *Search Configuration and Indexer Management* in this manual.

## 8.2 Indexer Services

---

The Indexer is represented by two CM services:

- **Index changes notifier**

Creates JMS (*Java Message Service*) messages with notifications when changes occur that concern the index. Stopping *index changes notifier* is ⚠ not ⚠ safe. If the Indexer module discovers that the notifier is stopped and there is a message that has to be sent to the persistent store, the Indexer will set the index status configuration property (*cmas-core-index-common, index.status*) to RED, i.e., signal that index needs full synchronization. Please see also section [System Monitoring, Checking the Status of the Indexer](#).

- **Index changes receiver**

Reads a JMS queue and starts update in Indexer. Stopping *index changes receiver* is safe. After restart it will pick up all of the missing changes from the persistent store (see section below about persistent store).

The services can be started/stopped using the Admin Tool. For details, please refer to the *ConSol CM Administrator Manual*, section *Services*.

## 8.3 Indexer Directory Structure

---

Please see section [ConSol CM File System Structure](#), [ConSol CM Data Directory](#).

## 8.4 Indexer Architecture and Update Principle

### 8.4.1 Basic Principle

The index is created, stored, and updated by CM on one server. In an environment with master and slave indexing servers, this is the master server. In single-server environments, the only existing CM server stores the index.

In the following section, the CM principle will be explained for the master and slave servers. For a single-server environment, the *master server* can be seen as synonymous with the *single server*.

Each server, master as well as slave servers, stores the index in its data directory (`<CMAS_DATADIR>/index/` directory, see previous section). The index is updated on the master server only. Each slave server polls the master server for updates. This implies a short delay between the update of the master index and slave indices.

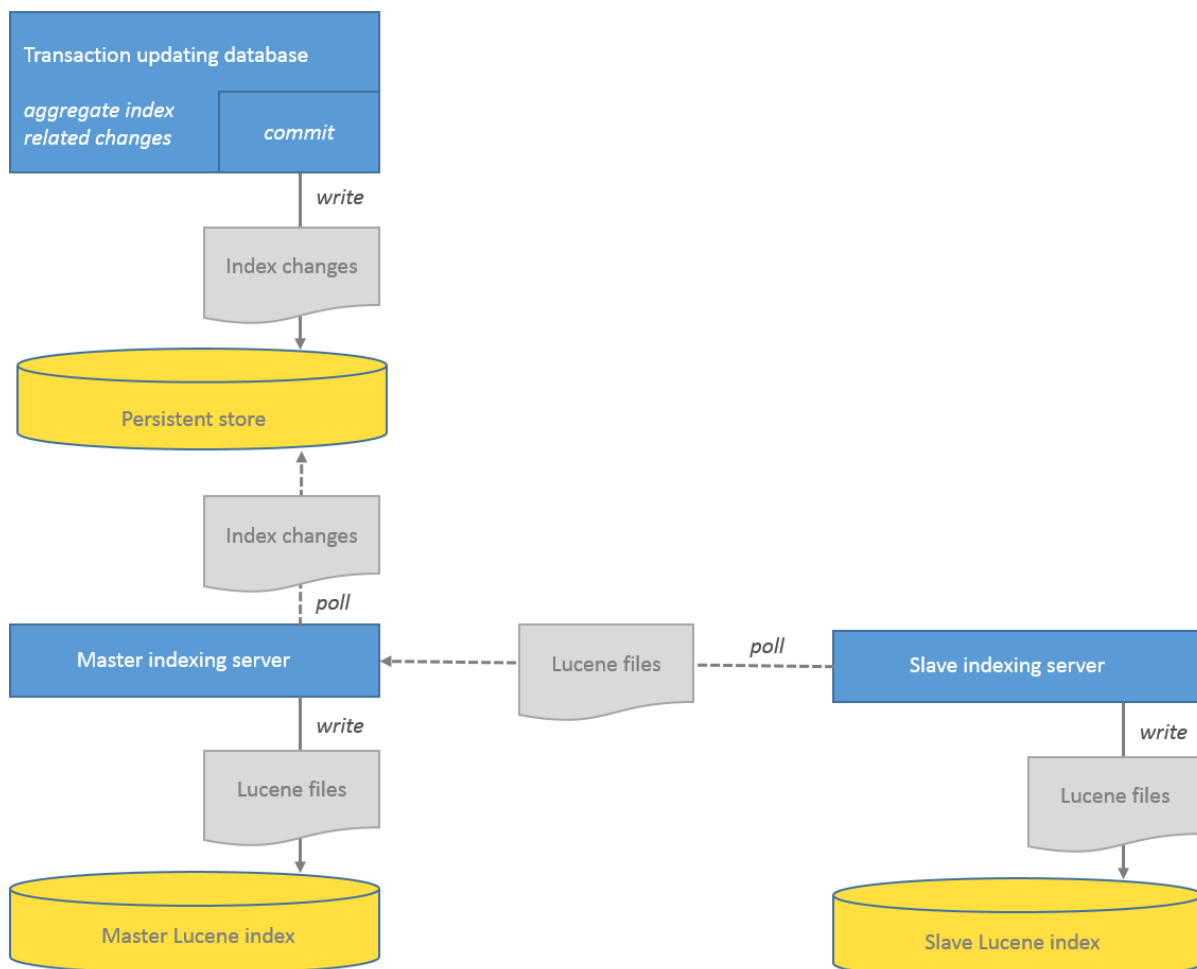


Fig. 1: ConSol CM - Indexer Architecture

## 8.4.2 Indexer Update, Step 1: Persistent Store Is Filled

When parameters in the system have changed, a transaction aggregates the IDs of affected entities and sends them to the **persistent store** at the transaction commit. The persistent store can be:

- a JMS queue: *queue/cm6-index*

or

- a database table: *cmas\_index\_update\_serialized*

Which of the two will be used is defined by the system property *cmas-core-index-common, database.notification.enabled*.

### Index Persistent Store as JMS Queue

*database.notification.enabled = false*

The JMS queue *queue/cm6-index* will be used.

### Index Persistent Store as Database Table

*database.notification.enabled = true* (not supported for CM versions before 6.9.4.1!)

The database table *cmas\_index\_update\_serialized* will be used for Indexer transactions (as persistent store). The master indexing server (or the only indexing server in single-server environments) polls this database.

## 8.4.3 Indexer Update, Step 2: Master Server Update

The master indexing server polls the persistent store for the index changes, aggregates them and stores the respective tasks in the database tables *cmas\_index\_update\_task* and *cmas\_index\_update\_part*. The master indexing server then runs the threads which execute *cmas\_index\_update\_task* and *cmas\_index\_batch\_update\_task* instances (i.e. tasks) to update the Lucene index files.

## 8.4.4 Administrative Changes

The following operations are considered *administrative changes*:

- Created automatically when one of the following was updated: scope, queue, enum value, ticket function, ticket engineer, supported locale, role, etc., if *No automatic commit of administrative changes* option is unchecked.
- Processed automatically if *No automatic commit of administrative changes* option is unchecked.
- Using *Commit administrative changes* command will start all administrative changes tasks.



Administrative changes are optionally stored within database table *cmas\_index\_administrative\_task* for deferred processing.

### 8.4.5 Indexer Update, Step 3: Slave Server Synchronization

Each slave indexing server polls the master indexing server for index updates. The master will send only the missing index files, most of the time these are very small, but on rare occasions one file may have up to 2 GB. Please note that the slave server first downloads the files into the *temp* directory, then - if download succeeded - the files are moved to the *index* directory. Be prepared to have a lot of free space both for the *index* and the *temp* directories. Having a *temp* directory on the same drive as the *index* directory will not copy files, but only change their location (which will be faster).

The *temp* directory is defined as follows: `System.getProperty("java.io.tmpdir")`.

### 8.4.6 Indexer Info about Manual Index Updates (Using the Admin Tool)



Please note that data in the index is always synchronized with the ConSol CM database, i.e. during an Index update no data is deleted/removed from the index files. The index is fully usable during the synchronization process, i.e. the search operations (detail as well as quick search) can be used with their full functionality and the complete data set. Changes made after the synchronization was started are immediately reflected in the index, because these data have a higher priority than the data which is synchronized due to an index update triggered manually using the Admin Tool.

When an admin has started the index update manually (Synchronize Index), all other index tasks are removed, before this update starts.

## 8.5 Indexer Restart

---

The master indexing server has to be started first.

## 8.6 Index Update Failure Scenarios

---

### 8.6.1 Master Failover

*Master failover* means, the current indexing master server does not work or is no longer available, and one of the slave servers is elected as new master. An arbitrary slave server is selected as new master based on access to the database.

If a master server failover will take place, depends on the value of the system property *cmas-core-index-common, synchronize.master.timeout.minutes*.

#### Master Failover Active

*synchronize.master.timeout.minutes* is not 0.

When the master server fails, a number of retries to reach the master server is performed by the slave servers for a certain period of time. The system property *cmas-core-index-common, synchronize.master.timeout.minutes* indicates this retry-time in minutes. After this period of time, a new master is elected.

Enabling master failover will result in a full index synchronization in case of the new indexing master gets elected. Full index synchronization means:

- The entire index is recreated.
- Before the start all other index tasks are removed.

#### No Master Failover

*synchronize.master.timeout.minutes* = 0

A new master will never be elected.

### 8.6.2 Failure of Update Task

Failed execution of the *cmas\_index\_update\_task* will create a new task *cmas\_index\_update\_task* with type *REPAIR*. Such task will wait for the administrator to run via the Admin Tool, navigation group *Services*, navigation item *Index-> Repair index*. Repair task existence will set the *YELLOW* index status configuration property. This can also be used for monitoring of the Indexer, please see section [System Monitoring, Checking the Status of the Indexer](#).

## 8.7 Monitoring the Indexer

---

This is treated in the section [System Monitoring, Checking the Status of the Indexer](#).

## 8.8 Index Backup and Restore

All ConSol CM indices are stored on the file system. However, you cannot just go ahead and copy the files and directories during system operation, because this might lead to inconsistent states (e.g., *.lock* files are used). That means, in order to make a backup of the CM indices, you have to follow some basic principles. Here we offer you a backup concept.

### 8.8.1 Index Backup

The correct way is to use the following *HTTP* request using basic authentication for global admin:

```
wget http://${indexing.master.host:port}/index/snapshot --user ${admin} --password ${password} -
Obbackup.jar
```

This will download the full index into a *backup.jar* file. The received full snapshot will have the timestamp of the moment when the command was executed. Please see the following code block for an example of the file content.

```
unzip -l backup.jar
Archive:  backup.jar
  Length      Date    Time    Name
-----
    158  2012-08-23  11:09  META-INF/...
   7739  2012-08-23  11:09  ticket/_82.fdt
    280  2012-08-23  11:09  ticket/segments_7y
  12293  2012-08-23  11:09  ticket/_82.frq
    123  2012-08-23  11:09  ticket/_82.nrm
  10577  2012-08-23  11:09  ticket/_82.prx
  21624  2012-08-23  11:09  ticket/_82.tis
   1366  2012-08-23  11:09  ticket/_82.fnm
    956  2012-08-23  11:09  ticket/_82.fdx
    308  2012-08-23  11:09  ticket/_82.tii
        2012-08-23  11:09  unit/...
        2012-08-23  11:09  engineer/...
-----
  84070                                49 files
```

### 8.8.2 Index Restore

In order to recover/restore, e.g., a corrupted ticket index you have to:

- Stop the CM master indexing server (the CM server has to be stopped, not only the Indexer services).
- Clean the master indexing server directory `<CMAS_DATADIR>/index/index.${number}/ticket/`.
- Clean the master indexing server directory `<CMAS_DATADIR>/index/index.${number}/ticket.uuid/`.
- Unpack the `backup.jar/ticket/*` files into the master indexing server directory `<CMAS_DATADIR>/index/index.${number}/ticket/`.
- (Re-)start the CM master indexing server (the CM server has to be re-started, not only the Indexer services).

In a cluster environment, the master indexing server will automatically synchronize the recovered index with the slave nodes.

In order to update the index for the data which has been generated during the time of the backup, use the Admin Tool, navigation group *Services*, navigation item *Index* -> *Synchronize index*.

## 8.9 System Properties Which Are Relevant for the Indexer Master/Slave Synchronization

---

Please refer to the *ConSol CM Administrator Manual, Appendix D (Important System Properties, Ordered by Area of Application), Indexer & Search Configuration - Indexer*. The relevant properties start with *synchronize*. ...

## 8.10 JBoss Parameters Which Are Relevant for the Indexer Master/Slave Synchronization

---

One JBoss parameter which has an impact on Master/Slave synchronization is the HTTP header size ( `maxHttpHeaderSize`, `MAX_HEADER_SIZE`). This is because the list of files which have to be synchronized is transferred between Master and Slave in the HTTP header. If the maximum header size is too small, the communication between Master and Slave will break down. If the value of `MAX_HEADER_SIZE` is not specified, this attribute is set to 8192 (8 KB).

For ConSol CM, this value is only relevant in clustered environments, which are run as managed domain. Therefore, you have to set this value in the *domain.xml* file.

Example for setting a HTTP header size in JBoss config file

```
<system-properties>

    <property name="org.apache.coyote.http11.Http11Protocol.MAX_HEADER_SIZE" value="65535"/>

</system-properties>
```



## **9 ConSol CM Operations Manual - DWH Management**

---

## 9.1 DWH Management

---

- [Introduction](#)
- [Short Overview of CM Components Relevant for Reporting with CM](#)
  - [Basic Principle](#)
  - [DWH-Specific Log Files](#)
  - [Open DWH Tasks](#)

### 9.1.1 Introduction

If the BI/reporting functionality is active in your CM infrastructure, you will have to take care of two main components:

- The ConSol CM Reporting Framework (CMRF)
- The ConSol CM Data Warehouse (DWH)

A short overview of a system with reporting functionalities is provided in the current manual in section [Architecture of a CM System with CMRF and DWH](#). In case you need to know technical details about the CMRF/DWH set-up, please read the respective sections in the *ConSol CM Set-Up Manual*.

CMRF/DWH administration using the Admin Tool is explained in great detail in the *ConSol CM Administrator Manual*, section *Data Warehouse (DWH) Management*. In order to understand reporting in CM, we recommend to read this section first.

A quite detailed explanation of all processes of CM/CMRF/DWH synchronisation is provided in the current manual, in section [CMDB / CMRF/ Data Warehouse Synchronization Process](#).

### 9.1.2 Short Overview of CM Components Relevant for Reporting with CM

#### Basic Principle

The ConSol CM database and the DWH are two separate databases (or database schemas, depending on the RDBMS flavor). The CMRF, a Java EE application, is responsible for transferring the required data. This can be done using JMS queues or using direct database access. Please read the section *ConSol CM Administrator Manual*, section *Data Warehouse (DWH) Management*, subsection *Transfer Modes: JMS or DIRECT* for a detailed explanation.

In order to run smoothly, the following components have to be up and running:

1. The CM server has to run and to write data into the CM database.
2. The CMRF has to run and has to have access to the DWH database (schema).

3. The DWH mode has to be set to one of the following values:

a. **ADMIN**

This means that there is no automatic transfer of data from CM to the DWH. The administrator has to start the transfer manually.

b. **LIVE**

This means that the transfer of data from CM to the DWH is performed automatically, on-the-fly.

## DWH-Specific Log Files

In order to read the DWH-specific log files, you can either access the file system of the server directly (see section [ConSol CM Logging and Log Files](#)) or you can use the Admin Tool. For the latter, open the navigation group *Data Warehouse*, navigation item *DWH Configuration and Logs*. The content of the log file will be displayed in the main info box. Select the desired log files using the radio buttons in the upper right:

- Initialization
- Transfer
- Update

You can also check the original log file under the following path:

- **JBoss 5:**

```
<JBoss_HOME>\server\<CMRF_SERVER_NAME>\log\cmrf.log
```

- **JBoss 7 (single server):**

```
<JBoss_HOME>/standalone/log/cmrf.log
```

- **WebLogic:**

```
<WLS_HOME>\Middleware\user_projects\domains\consolcm6_domain\cmrf-logs\cmrf.log
```

Please note that these are the standard paths. In ConSol CM, e.g., *Log4J* is used (for JBoss 5 and Weblogic). They may be configured to use different paths in the *log4j.xml* file. A detailed description of CM logging and log files is provided in the section [ConSol CM Logging and Log Files](#).

Usually the log file and/or log panel entries give good hints regarding the initial reason for a transfer failure. If you run into a problem you cannot resolve and you have a maintenance contract with ConSol, please contact our support team.

## Open DWH Tasks

Use the Admin Tool to see open DWH tasks. They will be displayed in the navigation group *Data Warehouse*, navigation item *DWH Tasks*. In the CM database, the tasks are listed in the table *cmas\_dwh\_task*.

In the list of DWH tasks, you will find entries (one entry per task) if ...

- the DWH is running in *ADMIN* mode and the administrator has started an update: all tasks that have to be performed are listed.
- the DWH is running in *LIVE* mode but the check box *Automatic commit of administrative changes* has not been checked.
- Custom Field, Data Object Group Field, or Resource Field annotations have been set to *reportable = true* and the checkbox *Automatic commit of administrative changes* has not been checked.

You can mark a task in the list and execute it manually.

If the checkbox *Automatic commit of administrative changes* has been checked, the tasks will be run automatically by the system.

## 9.2 CMDB / CMRF/ Data Warehouse Synchronization Process

---

- [CMDB / CMRF/ Data Warehouse Synchronization Process](#)
  - [Introduction](#)
  - [CM / CMRF / DWH Synchronization: General Principle](#)
    - [Intention when Using a DWH](#)
    - [ConSol CM System Architecture with CMRF and DWH](#)
    - [Synchronization Mechanism](#)
      - [Synchronization Mode](#)
      - [Transfer Mode](#)
      - [Components Involved in Transfer in DIRECT Mode](#)
        - [Tables in the CM Database \(or Database Schema\)](#)
        - [Tables in the DWH \(Database or Database Schema\)](#)
        - [CM Processes and Services](#)
  - [Synchronization Procedures](#)
    - [Operations Triggered Using the Admin Tool](#)
      - [Admin Tool: Initialize Button \(without Overwrite\)](#)
      - [Admin Tool: Initialize Button \(with Overwrite\)](#)
      - [Admin Tool: Transfer Button](#)
      - [Admin Tool: Update Button](#)
      - [Admin Tool: Enabling LIVE Mode \(OFF/ADMIN -> LIVE\)](#)
      - [Admin Tool: Run tasks Button](#)
    - [Operations performed by CM and by DWH Services](#)
      - [DWH Transfer Service](#)
      - [CMRF Service](#)
        - [TRANSFER Data](#)
        - [LIVE Data](#)
        - [CONTROL Data](#)
      - [DWH Log Service](#)
      - [CM in LIVE Mode](#)
      - [DWH Live Service](#)
    - [Operations via JMX](#)
      - [JMX \(CM6\)](#)
      - [JMX \(CMRF\)](#)
- [CM / CMRF / DWH Synchronization: FAQs and Tips for Troubleshooting](#)
  - [Question #1: DWH Tasks](#)
  - [Question #2: DWH Update Time](#)
  - [Question #3:CMRF/DWH Monitoring](#)
  - [Question #4: LIVE Mode Specialties](#)
  - [Question #5: Manual Submission of Tasks in LIVE Mode](#)
  - [Question #6: RecoverableExceptions](#)
  - [Question #7: Debug Level for CMRF](#)

## 9.2.1 Introduction

As a CM operator or administrator, you might have to solve problems with the Data Warehouse (DWH) in case the synchronization from the CM database to the DWH does not run smoothly. This is why we provide detailed information about the DWH synchronization process, so you can base your error analysis and troubleshooting on this information.

Please also read the *ConSol CM Administrator Manual*, section *Using CM for Reporting - Data Warehouse DWH Management* for a detailed introduction to the CMRF/DWH administration. In the current section, we assume that you know how to administer the DWH.

## 9.2.2 CM / CMRF / DWH Synchronization: General Principle

### Intention when Using a DWH

A DWH provides several advantages, the most prominent ones are:

- no need to use the production database for reporting, thus avoiding performance decrease
- a database schema and table structure which is optimized for reporting-typical database queries

A detailed introduction to this topic is provided in the *ConSol CM - Data Warehouse (DWH) Documentation*.

## ConSol CM System Architecture with CMRF and DWH

This is explained in detail in section [Architecture of a CM System with CMRF and DWH](#).

## Synchronization Mechanism

### Synchronization Mode

There are three different values for the synchronization mode for transferring data from ConSol CM to a DWH database. This is the value of the CM system property *cmas-dwh-server,dwh.mode*.

- **OFF**  
No data transfer to the DWH.
- **LIVE**  
In this mode, every change that is submitted to the ConSol CM database is immediately synchronized with the DWH.
- **ADMIN**  
In this mode, the administrator has to trigger the synchronization manually.

**i** Always use the graphical configuration (navigation group *Data Warehouse*, navigation item *DWH Configuration and Logs*, button *Configuration*) to set the synchronization mode! Setting it directly in the CM system properties does not work!

## Transfer Mode

The synchronization mechanism depends on the transfer mode:

- JMS mode
- DIRECT mode

Both modes are explained in the *ConSol CM Administrator Manual*, section *Using CM for Reporting - Data Warehouse DWH Management*. Starting with CM version 6.9.4.1, the DIRECT mode is the default value. Starting with CM version 6.11.0, the DIRECT mode is the only possible mode (JMS is no longer available). This is why in the current section, only the DIRECT mode is explained.

## Components Involved in Transfer in DIRECT Mode

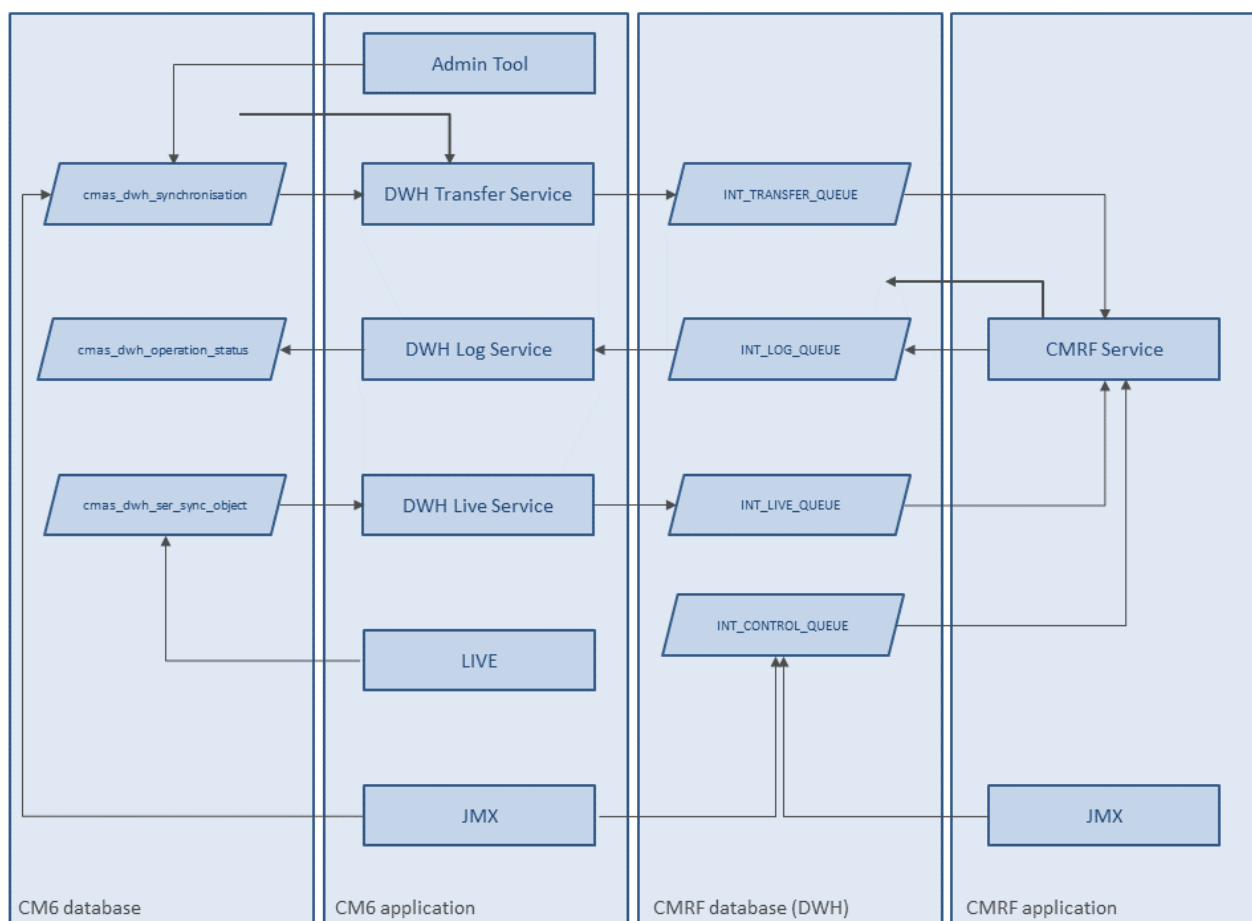


Fig. 1: Services and tables involved in CM/CMRF/DWH data transfer (DIRECT mode)

### Tables in the CM Database (or Database Schema)

- **cmas\_dwh\_operation\_status**  
The DWH Log Service writes into this table to document the DWH status.
- **cmas\_dwh\_ser\_sync\_object**  
This table is only used in LIVE mode. CM inserts an entry into this table for a transaction which was performed in the CM database. The entry contains a serialized package which needs to be sent to CMRF. The DWH Live Service reads from this table. See details below.
- **cmas\_dwh\_synchronization**  
When a DWH operation is performed using the Admin Tool (see details below), an entry is created in this table. The column *type* contains one of the possible values: INITIALIZATION, REINITIALIZATION, TRANSFER, UPDATE, TASK. The DWH Transfer Service reads from this table. See details below.
- **cmas\_dwh\_task**  
An entry in this database is created when an annotation which is relevant for DWH transfer has been changed, e.g. *reportable group*, *reportable*, *dwh-no-history*, *dwh-no-history-field*. Tasks listed in this table are visible in the Admin Tool (navigation group *Data Warehouse*, navigation item *DWH Tasks*). When a task is started (automatically or manually), a new entry in *cmas\_dwh\_synchronization* is created. When the data has been processed successfully, the entry from *cmas\_dwh\_task* is deleted.

### Tables in the DWH (Database or Database Schema)

- **INT\_CONTROL\_QUEUE**  
INT\_CONTROL\_QUEUE is used to control the processing in CMRF. Two actions are available: *pause* and *resume*. Only the [JMX components](#) write into INT\_CONTROL\_QUEUE. Only the CMRF service reads and deletes entries from INT\_CONTROL\_QUEUE. It is done during processing of other data (e.g. transfer). The CMRF service pauses processing after *pause* action and continues after *resume* action.
- **INT\_LIVE\_QUEUE**  
The DWH Live Service writes serialized packages into this table. CMRF reads from this table to process the LIVE data.
- **INT\_TRANSFER\_QUEUE**  
The DWH Transfer Service writes serialized packages into this table. CMRF reads from this table to process the TRANSFER data.
- **hlp\_parameter**  
This table is not directly involved in CM/CMRF/DWH transfer but provides some useful information. The most important value in this context is the *parameter\_name dwh\_status*. The possible values are explained in the following table.

Value of dwh_status (ID)	Description of the value	Explanation
--------------------------	--------------------------	-------------



Value of dwh_status (ID)	Description of the value	Explanation
0	DWH uninitialized	This is the status after application startup, when DWH sees that DB has not been initialized (e.g. no entries in static data). This is a theoretical value, because when DWH is uninitialized, the DWH does not yet exist.
1	DWH waiting for initial transfer	DWH waiting for initial transfer (ID=1) is set during (re) initialization. This is the status after DWH application startup when DWH sees that there has not been a data transfer yet. When action transfer sends transfer messages, DWH switches to status <i>DWH: data transfer in progress</i> (ID=2).
2	DWH: data transfer in progress	CMRF is still working on transfer messages. In this status, DWH ignores all incoming non-transfer messages. There may have been problems with the transmission. DWH enters this state when action transfer is executed. It remains in this state until either a fatal error occurs (then state is switched to <i>DWH: data transfer finished unsuccessfully</i> (ID=3)) or all transfer messages have been handled (in which case the subsequent status is either ( <i>DWH: data transfer finished unsuccessfully</i> (ID=3) when there were errors, or <i>DWH operational</i> (ID=4) when there were no errors).

Value of dwh_status (ID)	Description of the value	Explanation
3	DWH: data transfer finished unsuccessfully	All data transfer messages have been handled, however, there were some errors.
4	DWH operational	DWH is operational and will handle all incoming update messages from CM. This status is reached when all data transfers or updates have finished successfully.

- **hlp\_transfer\_error**: Keeps a copy of the error message from exceptions of CMRF transfer since the last REINITIALIZE. Here you can look up the *object\_uid* for which the exception occurred.

## CM Processes and Services


The following processes and services are relevant for the CMDB-DWH data transfer. The details are explained in the following sections.

- CM in LIVE mode
- DWH Live Service
- DWH Log Service
- DWH Transfer Service
- CMRF Service



## Synchronization Procedures

The synchronization process can be started by a manual action, i.e. when the administrator clicks on a button in the Admin Tool, starting a DWH action, or the sync process can be triggered automatically in LIVE mode. Please compare [the figure explaining all parts of the synchronization processes](#).

## Operations Triggered Using the Admin Tool

 A DWH operation which is performed using the Admin Tool does not do anything but write one or more entries into the CM table *cmas\_dwh\_synchronization*!. Tasks in this table are executed according to the timestamp of their creation in ascending order.

So please note that if you press a button several times, a new entry in *cmas\_dwh\_synchronization* is created with each click! The respective operations will then have to be executed one after another, ordered by the timestamp of their creation. Thus, there is the risk of accumulating DWH operations which might decrease CM performance considerably and might produce unwanted results.

Please wait until you see the result of an operation (Admin Tool is unblocked) and do  not  create a pipeline of pending operations which might interfere with one another.

**Example:** the admin clicks on *Initialize*, nothing happens (because the system is working behind the scenes preparing the initialization). The admin clicks on *Update*. Nothing happens. The admin clicks on *Initialize* again ...

What will happen?

Three entries have been created in the CM table *cmas\_dwh\_synchronization*. (1) INITIALIZE, (2) UPDATE, (3) INITIALZE

The initialization (1) is performed. When this is done, the respective entry is deleted from *cmas\_dwh\_synchronization*.

The update (2) is performed. When this is done, the respective entry is deleted from *cmas\_dwh\_synchronization*.

Now you could start work with the DWH, but ...

The next initialization (3) is performed. When this is done, the respective entry is deleted from *cmas\_dwh\_synchronization*.

The DWH is now in a non-operable state, and a transfer or update is required.

### Admin Tool: Initialize Button (without Overwrite)

A new entry in *cmas\_dwh\_synchronization* is created (type='INITIALIZATION', comment\_='AT - INITIALIZE', dwh\_status='NEW', cmrf\_status='NEW', next\_serial\_number=0, creation\_date=<current date>). The initialization is then processed in another thread (by DWH Transfer Service). The Admin Tool will be unblocked when the processing of the initialization is finished in CM6 (*cmas\_dwh\_synchronization.dwh\_status*='SUCCESS'|'ERROR').

**Admin Tool: Initialize Button (with Overwrite)**

A new entry in *cmas\_dwh\_synchronization* is created (type='REINITIALIZATION', comment\_='AT - REINITIALIZE', dwh\_status='NEW', cmrf\_status='NEW', next\_serial\_number=0, creation\_date=<current date>). The reinitialization is then processed in another thread (by DWH Transfer Service). The Admin Tool will be unblocked when the processing of the reinitialization is finished in CM6 (*cmas\_dwh\_synchronization.dwh\_status*='SUCCESS'|'ERROR')

**Admin Tool: Transfer Button**

A new entry in *cmas\_dwh\_synchronization* is created (type='TRANSFER', from\_date=0, to\_date=<current date>, comment\_='AT - TRANSFER', dwh\_status='NEW', cmrf\_status='NEW', next\_serial\_number=0, creation\_date=<current date>). The transfer is then processed in another thread (by DWH Transfer Service). The Admin Tool will be unblocked when the processing of the transfer is finished in CM6 (*cmas\_dwh\_synchronization.dwh\_status*='SUCCESS'|'ERROR').

**Admin Tool: Update Button**

A new entry in *cmas\_dwh\_synchronization* is created (type='UPDATE', from\_date=null, to\_date=<current date>, comment\_='AT - UPDATE', dwh\_status='NEW', cmrf\_status='NEW', next\_serial\_number=0, creation\_date=<current date>). The update is then processed in another thread (by DWH Transfer Service).

The Admin Tool will be unblocked when the processing of the update is finished in CM6 (*dwh\_status*='SUCCESS'|'ERROR').

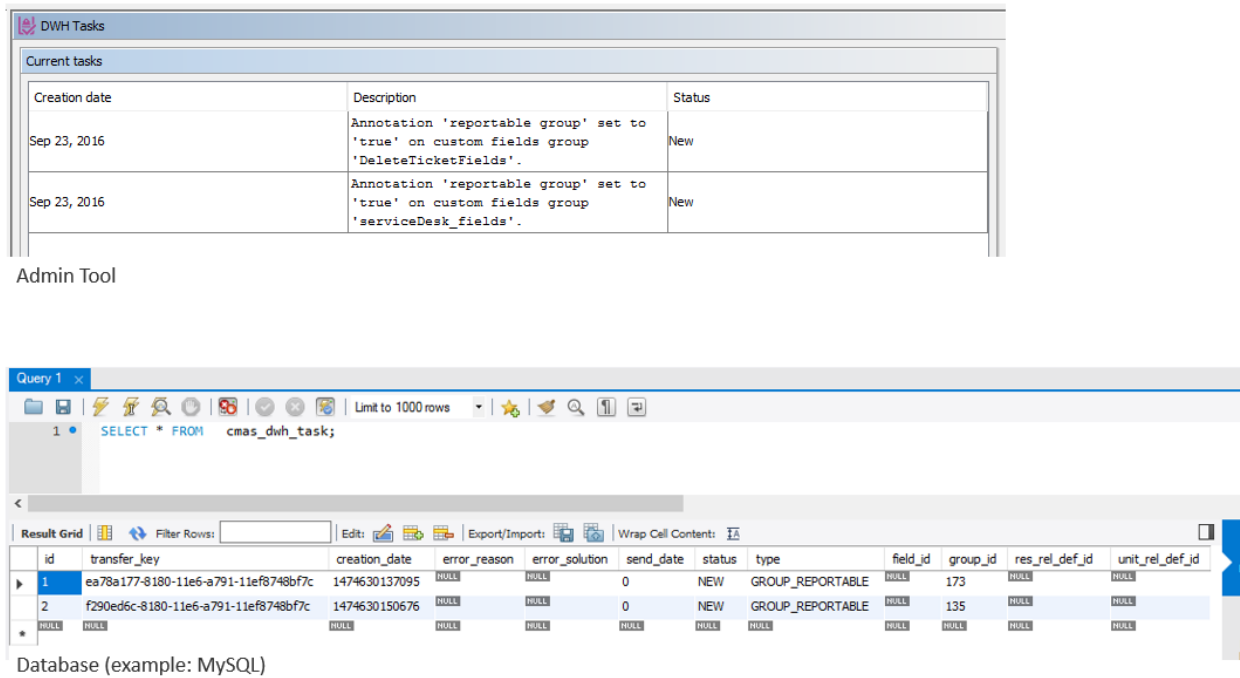
**Admin Tool: Enabling LIVE Mode (OFF/ADMIN -> LIVE)**

When the synchronization mode is set to LIVE using the Admin Tool (navigation group *Data Warehouse*, navigation item *DWH Configuration and Logs*), the CM system property *cmas-dwh-server.live.start* is created and set to the current date.

If an update is needed, a new entry in *cmas\_dwh\_synchronization* is created (type='UPDATE', from\_date=null, to\_date=<current date + value of property 'time.buffer'>, comment\_='Live mode automatic update', retry\_count=3, dwh\_status='NEW', cmrf\_status='NEW', next\_serial\_number=0, creation\_date=<current date>). The update is then processed in another thread (by DWH Transfer Service).

**Admin Tool: Run tasks Button**

This button is available/active if there are tasks which have not yet been performed. They are stored in the CM table *cmas\_dwh\_task* and are displayed in the Admin Tool (navigation group *Data Warehouse*, navigation item *DWH Tasks*).



The figure consists of two screenshots. The top screenshot shows the 'DWH Tasks' section in the Admin Tool. It displays a table with two rows of tasks, both with a status of 'New'. The bottom screenshot shows a SQL query result in a database (MySQL) for the table 'cmas\_dwh\_task'. The query is 'SELECT \* FROM cmas\_dwh\_task;'. The result grid shows two rows of data.

Creation date	Description	Status
Sep 23, 2016	Annotation 'reportable group' set to 'true' on custom fields group 'DeleteTicketFields'.	New
Sep 23, 2016	Annotation 'reportable group' set to 'true' on custom fields group 'serviceDesk_fields'.	New

Admin Tool

id	transfer_key	creation_date	error_reason	error_solution	send_date	status	type	field_id	group_id	res_rel_def_id	unit_rel_def_id
1	ea78a177-8180-11e6-a791-11ef8748bf7c	1474630137095	NULL	NULL	0	NEW	GROUP_REPORTABLE	NULL	173	NULL	NULL
2	f290ed6c-8180-11e6-a791-11ef8748bf7c	1474630150676	NULL	NULL	0	NEW	GROUP_REPORTABLE	NULL	135	NULL	NULL

Database (example: MySQL)

Fig. 2: Pending DWH tasks in Admin Tool and in CM database

When the admin has clicked the *Run task* button, a new entry in *cmas\_dwh\_synchronization* is created for each selected task (type='TASK', from\_date=null, to\_date=null, comment\_='AT-TASK executed manually', reference\_id=<id of the task>, dwh\_status='NEW', cmrf\_status='NEW', next\_serial\_number=0, creation\_date=<current date>). The tasks are then processed in another thread (by DWH Transfer Service).

A task can also be started automatically if the option *Automatic commit of administrative changes* is enabled.

## Operations performed by CM and by DWH Services

### DWH services information

When you operate CM in a cluster, each of the DWH services is active on only one node. This does not have to be the same node for each service. For example, DWH Transfer Service can run on cluster node 1 and DWH Live Service can run on cluster node 2. The services which are not needed can be stopped (using the Admin Tool) on the nodes where they would be inactive.

The DWH services only process data when the DWH mode (CM system property *cmas-dwh-server*, *dwh.mode*) = ADMIN or LIVE. If the DWH mode = OFF, nothing is processed.

## DWH Transfer Service

The DWH Transfer Service works in the same way in ADMIN and LIVE mode.

The DWH Transfer Service reads entries from the CM table *cmas\_dwh\_synchronization* (with *dwh\_status* = *NEW* or *ACTIVE*) and executes the respective jobs in the order of the timestamp of the entry (in ascending order). The service inserts serialized packages into *INT\_TRANSFER\_QUEUE*. *cmas\_dwh\_synchronization.dwh\_status* is changed during processing (*NEW* -> *ACTIVE* -> *SUCCESS*/*ERROR*). *dwh\_status* is set to 'ACTIVE' when the entry is processed (only one entry can be active). *dwh\_status* is set to 'SUCCESS' if the processing is finished successfully. *dwh\_status* is set to 'ERROR' if the processing is finished unsuccessfully.

During the ACTIVE period, the DWH Transfer Service inserts serialized packages into *INT\_TRANSFER\_QUEUE*. The package size can be defined using the CM system property *cmas-dwh-server.batch-commit-interval*. Each package has a serial number. When a package has been written to the *INT\_TRANSFER\_QUEUE*, the value of the next serial number (*cmas\_dwh\_synchronization.next\_serial\_number*) is incremented.

### Processing of one entry:

1. The entry is read from *cmas\_dwh\_synchronization* (*dwh\_status* = *ACTIVE* | *NEW*)
2. If *dwh\_status* = *NEW*, *dwh\_status* is updated (*dwh\_status* = *ACTIVE*)
3. Data are sent to CMRF in small packages. The size of the packages depends on the CM system property *cmas-dwh-server.batch-commit-interval*. (Can be added in the Admin Tool, type: Integer). Each package has a serial number.
  - a. Sending of one package:
    - i. A portion of data is read from the CM database (table *cmas\_dwh\_synchronization*).
    - ii. The package is created.
    - iii. The serialized package is inserted into the *INT\_TRANSFER\_QUEUE*. The value of the next serial number (*cmas\_dwh\_synchronization.next\_serial\_number*) is incremented. The progress of the processing (*cmas\_dwh\_synchronization.position*) is updated. These operations are done in one transaction.
4. Processing in CM is finished, *cmas\_dwh\_synchronization.dwh\_status* is updated (*dwh\_status* = *SUCCESS* | *ERROR*).

### Error handling:

After an exception the processing will be continued from the point where it was stopped (*cmas\_dwh\_synchronization.position*) if:

- the DWH Transfer Service is stopped
- a database failure is detected
- the exception is recoverable (CM system property *cmas-dwh-server.recoverable.exceptions*, see also [FAQ about recoverable exceptions](#))
- retry is configured (*cmas\_dwh\_synchronization.retry\_count* > 0)
- it is an error (e.g. *OutOfMemoryError*) - in this case the DWH Transfer Service is stopped

Otherwise the processing will not be continued (*dwh\_status* = *ERROR*).

## CMRF Service

CMRF reads and processes TRANSFER data (types *INITIALIZATION*, *REINITIALIZATION*, *TRANSFER*, *UPDATE*, *TASK*), LIVE data and CONTROL data (*pause* and *resume*). TRANSFER data have a higher priority than LIVE data. LIVE data are processed only if there are no TRANSFER data available.

- TRANSFER data are read from the INT\_TRANSFER\_QUEUE.
- LIVE data are read from the INT\_LIVE\_QUEUE.
- CONTROL data are read from the INT\_CONTROL\_QUEUE.

### TRANSFER Data

When the DWH Transfer Service has written entries into the INT\_TRANSFER\_QUEUE, the CMRF service can process these entries (CMRF service will also process other entries, this is treated below). The processing of each initialization/reinitialization/transfer/update/task is split into multiple transactions. The current state of the processing is saved in the database at the end of each transaction. After restart the state of the processing is loaded from the database and the processing is continued.

A package (BLOB entry in the *data* column) of the INT\_TRANSFER\_QUEUE table contains the new /modified object (e.g. engineer, unit, resource) in total, i.e. "The ticket <Number> now looks like this: <modified ticket data>".

### Error handling:

After an exception the processing will be continued from the point where it was stopped if:

- a database failure is detected - in this case processing is paused
- the exception is recoverable (system property *cmrf.recoverable.exceptions*)
- it is an error (e.g. OutOfMemoryError)

Otherwise the processing will be not continued.

### LIVE Data

When the DWH Live Service has written entries into the INT\_LIVE\_QUEUE, the CMRF service can process these entries.

### CONTROL Data

*Pause* and *resume* entries written by a JMX into the INT\_CONTROL\_QUEUE are read by the CMRF service.

## DWH Log Service

The DWH Log Service works in the same way in ADMIN and LIVE mode.

The DWH Log Service is responsible for providing log information to CM administrators (e.g., in the Admin Tool). The service reads serialized packages from INT\_LOG\_QUEUE and writes them into the CM table *cmas\_dwh\_operation\_status*. Entries in this table are displayed in the Admin Tool (navigation group *Data Warehouse*, navigation item *DWH Configuration and Logs*). For more information about this topic, please see section [Data Warehouse \(DWH\) Management](#).

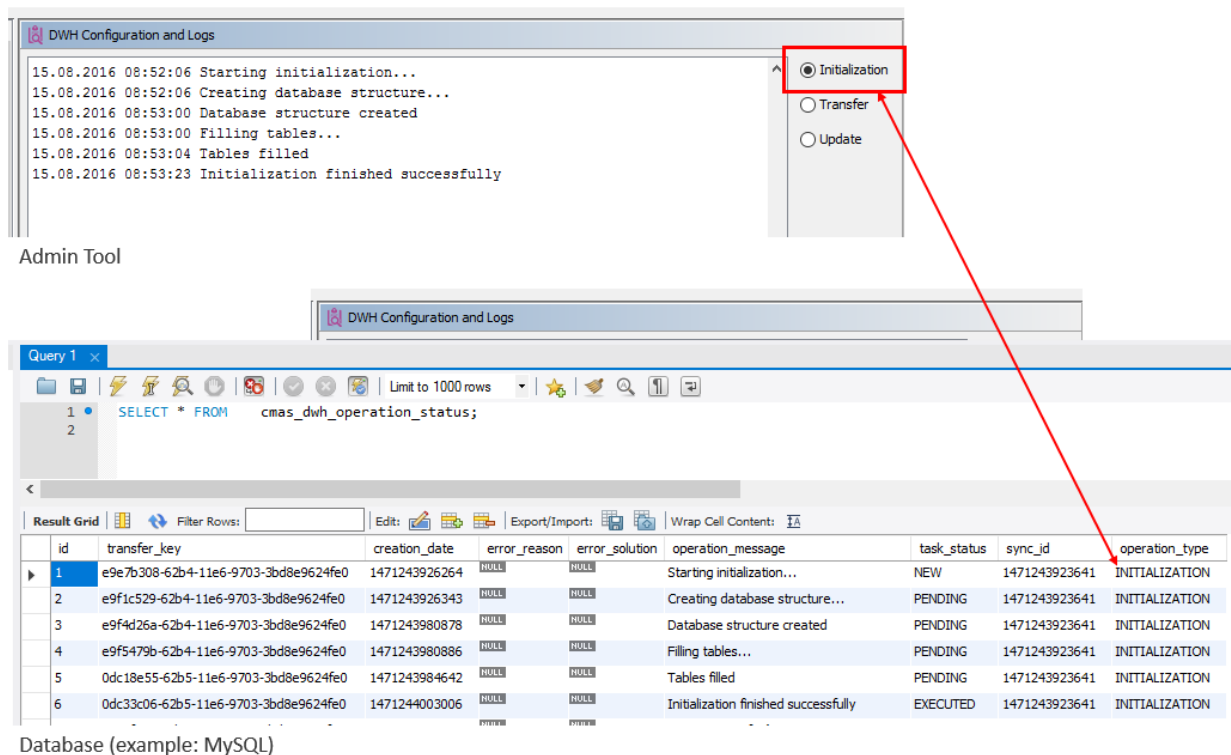


Fig. 3: DWH log entries in Admin Tool and database

One iteration (transaction):

1. Log entries are read and deleted from the INT\_LOG\_QUEUE.
2. A new entry in *cmas\_dwh\_operation\_status* is created for each log entry.
3. The property *last.success.live.timestamp* is updated for each LIVE log
4. *cmas\_dwh\_synchronization.cmrf\_status* is updated for each log entry except for LIVE log.

## CM in LIVE Mode

If LIVE mode is enabled and CMRF needs to be informed about changes done in a CM transaction, CM writes entries into the CM table *cmas\_dwh\_ser\_sync\_object*, one entry for each CM transaction which has been performed (e.g., a Custom Field for a ticket has been changed). The entry is written at the end of the transaction, before the *commit* operation is executed. Each entry contains a serialized package which needs to be sent to CMRF. The respective information will then be forwarded to CMRF by the DWH Live Service, see next paragraph.

## DWH Live Service

The DWH Live Service is only active if the synchronization mode is set to *LIVE* (i.e. if the CM system property *cmas-dwh-server, live.start* is set) and if all data older than the date of the property *live.start* were already sent to CMRF. If LIVE mode is not enabled and there is no data in *cmas\_dwh\_ser\_sync\_object*, the property *live.start* is deleted.



The DWH Live Service controls the just-in-time DWH update, i.e. it is responsible for sending information about changes of reportable data fields and data field groups in CM to CMRF. To do this, the DWH Live Service reads entries from the CM table *cmas\_dwh\_ser\_sync\_object* and inserts the respective serialized packages into the INT\_LIVE\_QUEUE in the DWH.

One iteration (transaction):

1. Packages are read from *cmas\_dwh\_ser\_sync\_object* (max. 100, in order of creation).
2. The property *live.serial.number* is read.
3. A serial number is set for each package.
4. The packages are deleted from *cmas\_dwh\_ser\_sync\_object*.
5. The property *live.serial.number* is updated.
6. The serialized packages are inserted into INT\_LIVE\_QUEUE.

Packages from INT\_LIVE\_QUEUE will then be processed by DWH Transfer Service.

## Operations via JMX

### JMX (CM6)

"consol.cmas:type=admin,topic=global,name=dwh.synchronizationService" can start transfer and update - new entry in *cmas\_dwh\_synchronization* is created

"consol.cmas:name=cmrf.control" can pause and resume processing in CMRF - new entry in INT\_CONTROL\_QUEUE is created

### JMX (CMRF)

"consol.cmrf:name=cmrf.control" can pause and resume processing in CMRF - new entry in INT\_CONTROL\_QUEUE is created

## 9.2.3 CM / CMRF / DWH Synchronization: FAQs and Tips for Troubleshooting

### Question #1: DWH Tasks

**How can I delete tasks from the DWH Tasks list in the Admin Tool when the tasks have been performed but are still visible in the list? How do I know if the tasks are really finished?**

The task is deleted automatically when the processing in CMRF is finished successfully. You can see the status of processing for each task using this query:

```
select cmas_dwh_task.id, cmas_dwh_task.status, cmas_dwh_synchronization.dwh_status,
cmas_dwh_synchronization.cmrf_status from cmas_dwh_task left join cmas_dwh_synchronization on
cmas_dwh_task.id=cmas_dwh_synchronization.reference_id;
```

### Question #2: DWH Update Time

**On large installations, a DWH update can take quite long. What can I do to reduce the time?**

Set the CM system property *cmas-dwh-server, batch-commit-interval* (type: Integer) to a value not too small. We recommend to start with a value of 150. However, the optimum chunk size depends on the system performance. If the system has enough RAM, the value can be 1000 or even more. This will speed up the transfer considerably.

### Question #3: CMRF/DWH Monitoring

Please see the [CMRF/DWH paragraph in the System Monitoring section](#).

### Question #4: LIVE Mode Specialties

**When the DWH had to be recovered from a full back-up by a DBA, how can I restart the automatic transfer in LIVE mode without starting with reinitialization and full transfer?**

Probably it will cause an ERROR in the log file: "An error occurred. Update is needed. Live packages will be processed again after update."

and many WARNINGS like this: "Update is needed. Live packages will be processed again after update. ('from' date must be equal or lesser than X, 'to' date must be equal or greater than Y"

Solution:

1. Execute the following command:

```
update cmas_dwh_synchronization set cmrf_status='ERROR' where to_date > X;
```

2. Run the update (*Update* button in the Admin Tool).

### Question #5: Manual Submission of Tasks in LIVE Mode

**When the option *Automatic commit of administrative changes* is not active and I submit a task manually (using the GUI of the Admin Tool), which consequences does this have in LIVE mode?**

A task is created automatically when a configuration change has been submitted (e.g. set a Custom Field from *reportable = false* to *reportable = true*). A new entry is created in the table *cmas\_dwh\_task*, and the task is displayed in the list of DWH tasks in the Admin Tool. When you click on *Run task* in the Admin Tool, a new entry is created in the table *cmas\_dwh\_synchronisation*. The DWH Transfer Service reads the data from *cmas\_dwh\_synchronisation* and writes chunks into INTTRANSFER\_QUEUE in the DWH. The CMRF service reads this data and also reads the live data from INT\_LIVE\_QUEUE. Since for CMRF, the TRANSFER data have a higher priority than the LIVE data, the TRANSFER data is processed first, i.e. the new data which results from the configuration change is processed first. Then the LIVE data processing is continued.

### Question #6: RecoverableExceptions

**How can I prevent the transfer to stop when an error occurs?**

The property *recoverable.exceptions* basically takes a comma-separated list of Java classes as value. All exceptions which are mentioned in the list will not cause the abortion of the DWH transfer, but CM/CMRF will continue the transfer and ignore the respective exception(s). As one value (or even the only value) in the list, you can provide the Java class name of an exception and add a '+' at the end. Then all exceptions which inherit from this class will be ignored. Example: *java.lang.Exception+*. In this case, all exceptions will be ignored, because *java.lang.Exception* is the parent class of all exceptions in Java. A regular expression can be provided as optional parameter. The RegEx is applied to the message of the exception as filter. The following characters '(', ':', and '\' in RegEx need to be escaped (' => '\(', ':' => '\:', '\' => '\\'). Example value: *java.sql.SQLRecoverableException,java.lang.RuntimeException+.\*T.{1\,2}T.\**

Please note that the value for recoverable exceptions has to be set at two locations:

1. For CM: as ConSol CM system parameter, using the Admin Tool: *cmas-dwh-server, recoverable.exceptions*.
2. For CMRF: as -D Java system property at CMRF startup: *-Dcmrf.recoverable.exceptions=...*

## Question #7: Debug Level for CMRF

**How can I increase the debug level for CMRF during troubleshooting?**

For a detailed explanation of the CM log files, please read section [ConSol CM Logging and Log Files](#).

The debug level for CMRF can be changed in the CM/CMRF config file

- JBoss: <JBOSS\_HOME>/jboss-<version>/standalone/configuration/cmrf.xml
- Weblogic: <DOMAIN\_HOME>\domains\consolcm6\_domain\log4j.xml

```
<logger category="com.consol.cmrf">
    <level name="INFO"/>
    <handlers>
        <handler name="CMRF_FILE"/>
        <handler name="ERROR_FILE"/>
    </handlers>
</logger>
```

Set the level from INFO to WARNING or DEBUG.

## 10 ConSol CM and LDAP Authentication

---

- [Introduction to ConSol CM with LDAP](#)
- [LDAP for Engineer Authentication in the Web Client](#)
- [LDAP for Customer Authentication in the Portal CM.Track](#)

## 10.1 Introduction to ConSol CM with LDAP

---

In a standard ConSol CM system, LDAP can be used for:

- Engineer authentication in the Web Client
- Customer authentication in CM.Track

There might be additional interfaces, e.g., an engineer import via ETL operation from an Microsoft Active Directory into the CM database, but use cases like that are rather system-specific and can therefore not be treated in a general CM manual. If you have to operate such a customized CM system, please ask your ConSol CM consultant. We will be happy to help you.

## 10.2 LDAP for Engineer Authentication in the Web Client

---

In ConSol CM the authentication of engineers, i.e., the users who work with the CM Web Client, can be performed using an LDAP look-up. The functionality is explained in great detail in the *ConSol Administrator Manual*, section *LDAP Authentication in the Web Client*.

## 10.3 LDAP for Customer Authentication in the Portal CM.Track

---

In ConSol CM the authentication of customers in the portal, i.e., the users who work with CM.Track, can be performed using an LDAP look-up. The functionality is explained in great detail in the *ConSol Administrator Manual*, section *CM.Track V1 - Authentication Modes for the Portal* and *CM.Track V2 - Authentication Modes for the Portal*.

## 11 ConSol CM License

---

- [General Information about Licenses in ConSol CM](#)
  - [Control of Consumed Licenses](#)
    - [Control of Consumed Licenses Using Log Files](#)
    - [Control of Consumed Licenses Using Database Queries](#)
      - [For CM Versions up to 6.9](#)
      - [For CM Versions 6.10 and Up](#)
- [Managing the ConSol CM License Using the Admin Tool](#)
- [Expert Information about Accessing Content of CM Licenses](#)



## 11.1 General Information about Licenses in ConSol CM

You will receive the license from your ConSol CM consultant or from the ConSol CM office. The license file is a plain text file which might look like the following example:

---

```
[PROCESS_DESIGNER]
contractParty = Demo-Lizenz_6-10_10_User
products = PROCESS_DESIGNER
version = 6.10
expirationDate = 30.09.2016
licenses = 5
signature = [REDACTED]

[TRACK_USERS]
contractParty = Demo-Lizenz_6-10_10_User
products = TRACK
version = 6.10
licensedPer = UNIQUE_USER
expirationDate = 30.09.2016
licenses = 1
signature = [REDACTED]

[REST_USERS]
contractParty = Demo-Lizenz_6-10_10_User
products = REST
version = 6.10
expirationDate = 30.09.2016
licenses = 10
signature = [REDACTED]

[ADMINTOOL_USERS]
contractParty = Demo-Lizenz_6-10_10_User
products = ADMIN_TOOL
version = 6.10
expirationDate = 30.09.2016
enabledModules = RESOURCE_POOL
signature = [REDACTED]

[CONCURRENT_USERS]
contractParty = Demo-Lizenz_6-10_10_User
products = WEB_CLIENT
version = 6.10
expirationDate = 30.09.2016
licenses = 10
signature = [REDACTED]

[TRACK]
contractParty = Demo-Lizenz_6-10_10_User
products = TRACK
version = 6.10
expirationDate = 30.09.2016
licenses = 200
signature = [REDACTED]
```

Fig. 1: License File (Example)

A ConSol CM license file contains entries for several modules. For each module, the number of valid licenses is indicated. For example, the following excerpt of a license file shows the Web Client, *CONCURRENT\_USERS* section. Ten licenses have been purchased.

```
[CONCURRENT_USERS]
contractParty = Demo-Licence ConSol
products = WEB_CLIENT
version = 6.9
expirationDate = 31.12.2014
licenses = 10
signature = XXX
```

A detailed explanation of the sections of a license file is provided in the *ConSol CM Administrator Manual*, section *License Management*.

ConSol CM works with concurrent users (sometimes also called floating licenses), i.e., the number of users who are logged in simultaneously is registered, no user names are checked. That means, the number of engineers who are managed in the Admin Tool (see section *Engineer Administration*) does not have to be identical to the number of Web Client licenses.

A license is consumed when the user logs in. The license is handed back to the server when the user session is terminated, i.e., when the user logs out or when the user session is terminated automatically by the server because the session timeout has been reached (see system property *cmas-core-server, server.session.timeout*).

## 11.1.1 Control of Consumed Licenses

### Control of Consumed Licenses Using Log Files

When an engineer session in the Web Client has started, the following line is printed into the *server.log* file (example with demo user):

```
2016-07-14 10:01:17,943 INFO [ sessionTimeoutEngineerLogger] [-] New session for engineer:
Susan has started. Session id: 24vb62d4-4999-11e6-9076-3756a64ecac4
```

When an engineer has logged out, the following line is printed into the *server.log* file (example with demo user):

```
2016-07-14 10:06:25,315 INFO [ sessionTimeoutEngineerLogger] [Susan-24db62d4-4999-11e6-9076-37
14a64ecac4] Session of engineer: Susan has ended. Reason: logout. Session id: 24db62d4-4999-11
e6-9076-3714a64ecac4
```

When a session has been automatically finished because of the timeout, the following line is printed into the *server.log* file (example with demo user):

```
2016-07-14 09:35:32,303 INFO [ sessionTimeoutEngineerLogger] [-] Session of engineer: Susan
has ended. Reason: timeout. Session id: 52b46c0f-4991-11e6-9076-3714a64ecac4
```

Please note that in the log files, it is not possible to distinguish Web Client and Admin Tool sessions, but you can distinguish the login names and "filter" for non-admin users.

## Control of Consumed Licenses Using Database Queries

The sessions are saved in the CM database table *cmas\_user\_session*. Thus, you can find out how many users are logged in using the following statements.

### For CM Versions up to 6.9

#### Web Client sessions:

```
SELECT count(auth_username) FROM cmas_user_session WHERE end_date is null and session_source = '
WEB_CLIENT';
```

#### CM.Track sessions:

```
select count(*) from cmas_user_session where end_date is null and session_type = 'CUSTOMER';
```

### For CM Versions 6.10 and Up

#### Web Client sessions:

```
SELECT count(auth_username) FROM cmas_user_session WHERE end_date = 0 and session_source = 'WEB
_CLIENT';
```

**CM.Track sessions:**

```
select count(*) from cmas_user_session where end_date = 0 and session_type = 'CUSTOMER';
```



The monitoring user will not consume a license! See section [System Monitoring, Monitoring User Configuration](#) for details.

## 11.2 Managing the ConSol CM License Using the Admin Tool

---

If the license has expired or will expire soon, you can import a new license file using the Admin Tool. Of course the Admin Tool will always start, even if the license has expired. In the Web Client, in CM.Track, and in the Process Designer, the login is not possible when the license has expired.

In the Admin Tool, open the navigation group *System*, navigation item *License*.



### Attention:

There is no *Back* button to undo changes with one click when you enter or delete text in the *License* field. If you accidentally change parts of the license, either

- close the Admin Tool **without** clicking *Save*. This will discard all changes you made to the license text. When you restart the Admin Tool afterwards, the license will be in the same condition as it was before you made the changes.

OR

- use the Reload button **without** clicking *Save* before. The "old" data will be reloaded.

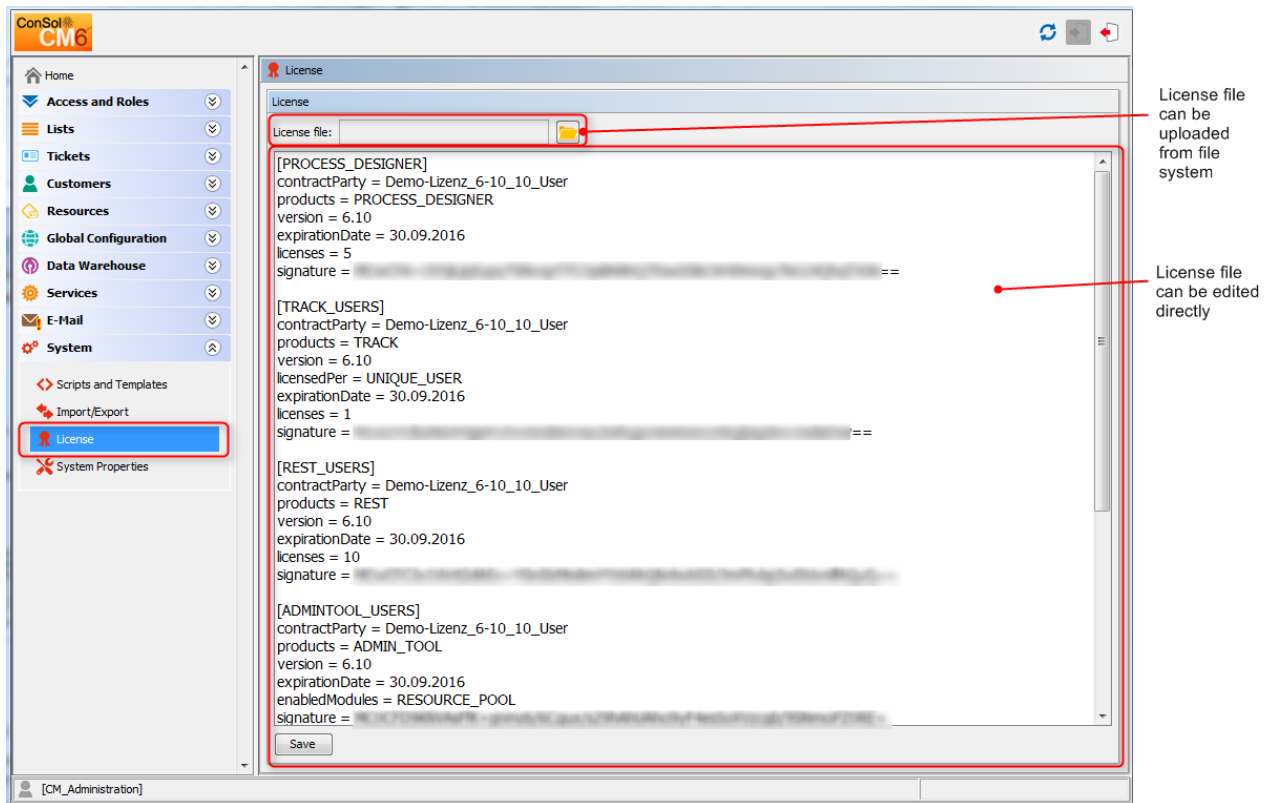


Fig. 2: ConSol CM Admin-Tool - System: License

Choose one of the two ways to import your ConSol CM license file:

- Insert the entire text of the license file by copy and paste. In case an old license is present, just replace the entire text. Click on *Save*.
- Load the license using the file browser next to the field *License file*. Click on *Save*.

You should receive a message that the license has been imported into the system successfully. It is in operation at once, without further action.

## 11.3 Expert Information about Accessing Content of CM Licenses

The content of CM licenses can also be queried using the MBean *licenceDeployer*. This MBean offers three methods, two for requesting license info and one to deploy the license:

- `getRemainingDays`
- `deployLicence`
- `getLicenceInfo`

If you are interested in using this feature and would like to have support implementing a system which uses the MBean access, e.g., to set up monitoring for your CM system, please contact your CM consultant.

The following figure shows an example using the JConsole.

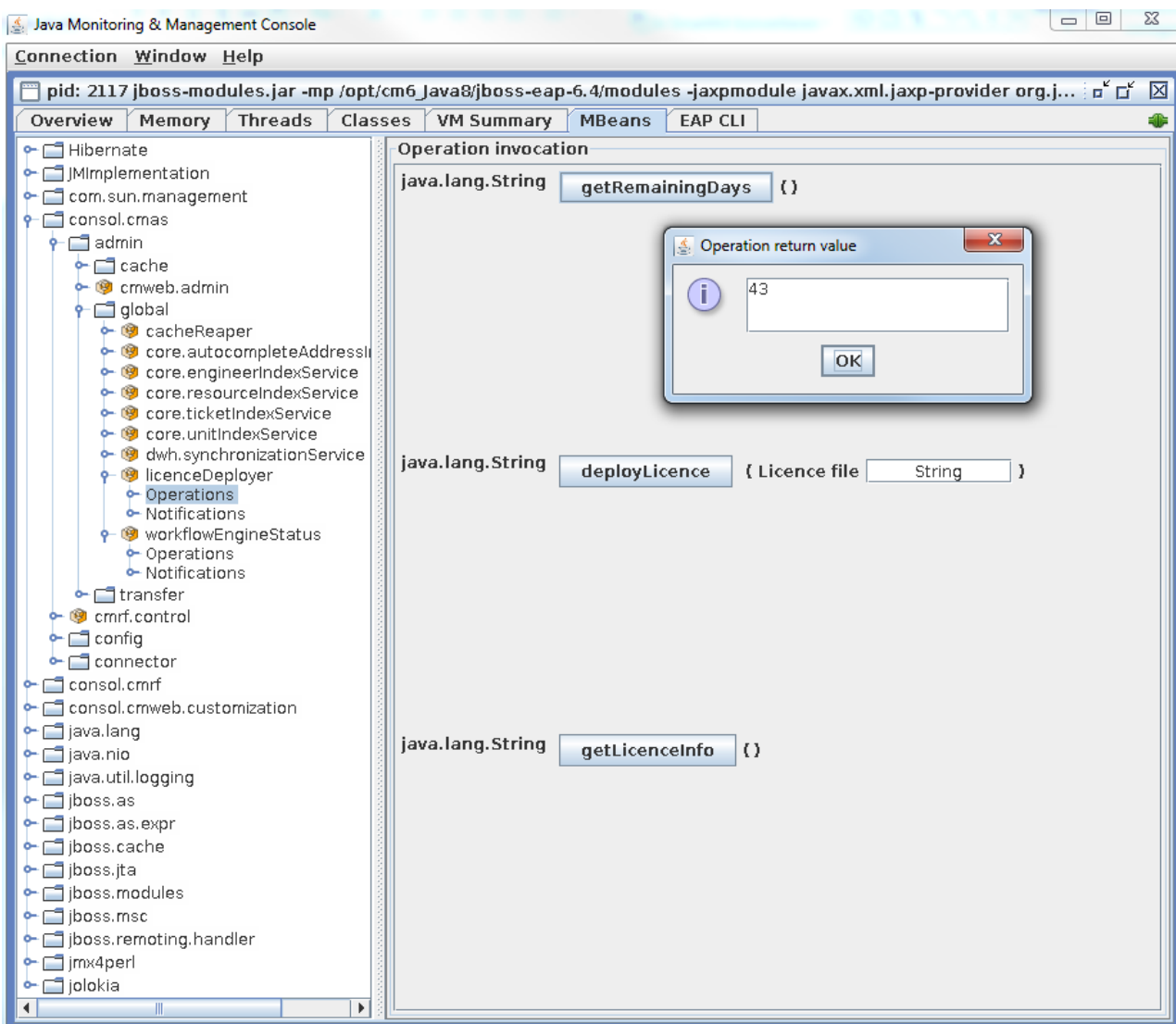


Fig. 3: Retrieving the Remaining Days of a CM License Using the JConsole



In case you have purchased a CM license with a limited period of validity, we recommend to set up a monitoring for the license which sends a notification a certain time before the license expires.



## 12 System Monitoring

---

- [Introduction](#)
- [Monitoring Tools](#)
- [Monitoring the ConSol CM Server Process](#)
  - [JMX Monitoring Using Jolokia](#)
- [Monitoring the Login into the CM Clients \(CM Web Client and CM.Track\), End-to-End Tests](#)
  - [Monitoring User Configuration](#)
  - [Web Client Monitoring Principle](#)
  - [Monitoring CM in a Cluster](#)
  - [URL /logout for Automation Purposes](#)
- [Using Beans to Monitor Some Basic Parameters](#)
- [Checking the Status of the Indexer](#)
  - [Checking the Indexer File System](#)
  - [Checking the Indexer Status in the Database](#)
- [Log File Monitoring](#)
  - [Tags to Monitor](#)
    - [Files and Tags Which Should Be Monitored](#)
- [Monitoring E-Mail Functionalities](#)
  - [CM Error E-Mail Configuration](#)
  - [Log File Control](#)
  - [Control of Undeliverable E-Mails](#)
- [CMRF/DWH Monitoring](#)

## 12.1 Introduction

---

You can use four "hooks" to monitor ConSol CM applications:

1. Check the **server process of the Java EE application**, i.e., the JBoss or WebLogic server process.
2. Check the **login of a monitoring ("dummy") user into the CM web clients**, i.e., the regular Web Client and the CM.Track client.
3. The **status of the Indexer** can be found out by controlling the file system and by retrieving a database parameter.
4. **Log file** monitoring.

## 12.2 Monitoring Tools

---

You can use a monitoring tool or application of your choice to control the CM systems. We recommend using a Nagios®-based solution. If you would like to get support on that topic, read the [ConSol Monitoring page](#) or ask your CM consultant.

## 12.3 Monitoring the ConSol CM Server Process

---

ConSol CM is a Java EE application, hence you can monitor the application server process. You can

- monitor the basic process parameters like CPU usage, see section [Using Beans to Monitor Some Basic Parameters](#)
- Monitor the CM functionality (is login possible?). See the following section.

### 12.3.1 JMX Monitoring Using Jolokia

You can use [Jolokia](#) to monitor application servers, e.g., memory usage and garbage collection, and you can also check ConSol CM functionalities. As a basis, you have to deploy the file *jolokia.war* into the application server. Please ask your ConSol CM consultant for support if you would like to use this option.

## 12.4 Monitoring the Login into the CM Clients (CM Web Client and CM.Track), End-to-End Tests

---

The default port for the CM web server is *8080*. That means, after a CM installation, you can reach the ConSol CM start page under *http(s)://<Server>:8080*. The CM Web Client can be reached under *http(s)://<Server>:8080/client/login*. Of course, if the port has been modified (e.g., by using a port offset), the ports in the monitoring scripts have to be adapted accordingly.

In order to check the client login, you have to create a monitoring user, e.g., *nagios* in the CM system. This user is created like a regular system user. Use the Admin Tool for this operation. For a detailed introduction to user, i.e., engineer administration, please refer to the *ConSol CM Administrator Manual*, section *Engineer Administration*.

### 12.4.1 Monitoring User Configuration

Starting with CM version 6.9.3, there is the option to configure a user (engineer/unit) for monitoring CM operations. This user can access each client exactly once using one session. This login will not consume a license. The session created will be marked as monitoring session. The user must independently have proper permissions to perform the tasks required for monitoring. These could include usage of the Admin Tool.

Two configurations have to be performed:

- **For the check of the Web Client:**  
Create an engineer and enter his login name in the system property *cmas.core.server, monitoring.engineer.login*.
- **For the check of CM.Track:**  
Create a contact, assign a CM.Track user profile to him (see section *CM.Track V1* or *CM.Track V2* in the *ConSol CM Administrator Manual*), and enter his login name into the system property *cmas.core.server, monitoring.unit.login*.

### 12.4.2 Web Client Monitoring Principle

1. Check the login  
For the login, the username and password have to be submitted using an HTML POST request.
2. E.g., call the CM main page

You might want to use the ConSol CM Nagios Plugin (a PERL script for Nagios servers) to check the Web Client. Please ask your ConSol CM consultant for more information.

## 12.4.3 Monitoring CM in a Cluster

In a cluster which consists of a load balancer and several CM servers, we recommend to check the login into each of the CM servers as well as the login via load balancer.

## 12.4.4 URL /logout for Automation Purposes

A special URL in the Web Client is being introduced with CM version 6.9.4:

```
http://<CM6_SERVER>/cm-client/logout/
```

This URL eases automation use cases like monitoring that require login and logout to the Web Client. It provides a robust way to log out of an automated Web Client session again not to unnecessarily consume licenses for monitoring.

## 12.5 Using Beans to Monitor Some Basic Parameters

To monitor basic application server parameters, e.g., Heap, CPU usage or used memory, you can use tools which directly access the JBeans of the application server. As a default in JBoss, use *JConsole*.

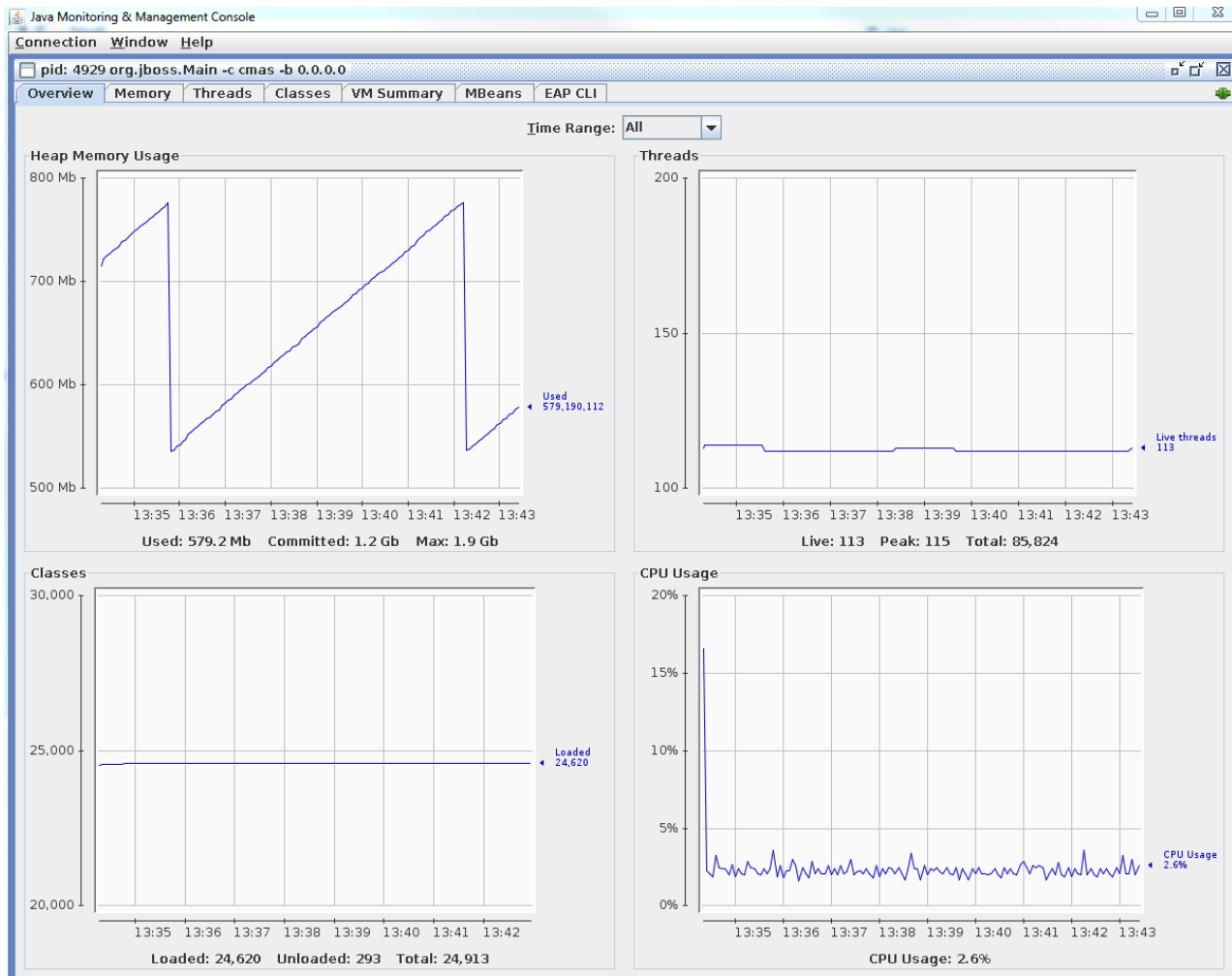


Fig. 1: JConsole - Checking Basic Runtime Values for a JBoss CM Server

In order to integrate this monitoring into a larger IT environment, you could use, e.g., [Jolokia](#) to receive HTML output from the request to the JBeans. This can be inserted into your monitoring environment, e.g., into Nagios using a Perl script.

## 12.6 Checking the Status of the Indexer

In order to perform effective searches, ConSol CM creates an index for each data field which is either marked as indexed by default or which has been marked as indexed by an administrator. A detailed introduction to the search and Indexer is provided in the *ConSol CM Administrator Manual*, section *Search Configuration and Indexer Management*. The indices are stored in the file system, the Indexer status is represented by a system property (*cmas-core-index-common, index.status*).

### 12.6.1 Checking the Indexer File System

The index is located in the data CM directory which has been configured during set-up, subdirectory *index*. You should ...

- control the disk space on this file system.
- control if the file system is available, in case it is mounted from another server.
- control if the synchronization between master and slave servers works in case you work in a server cluster environment.

### 12.6.2 Checking the Indexer Status in the Database

In order to control the Indexer status manually, start the Admin Tool and open the navigation group *Services*, navigation item *Index*. A detailed explanation is given in the *ConSol CM Administrator Manual*, section *Search Configuration and Indexer Management*.

To control the status using a monitoring tool, you can check the table *cmas\_configuration*. The system property *cmas-core-index-common, index.status* should be *GREEN*.

```
mysql> select * from cmas_configuration where property_ = 'index.status';
```

module_	property_	type_	description_	restart_	value_	system_	optional_
cmas-core-index-common	index.status	STRING	NULL	N	GREEN	Y	N

1 row in set (0.00 sec)

Fig. 2: Checking the Indexer Status Using Database Access

This property can also be seen in the Admin Tool, navigation group *System*, navigation item *System Properties*.

In case the status is *YELLOW* or *RED*, you have to use the Admin Tool to rebuild, repair, or refresh the index.



## 12.7 Log File Monitoring

---

For a detailed description of CM log files, please read section [ConSol CM Logging and Log Files](#).

### 12.7.1 Tags to Monitor

#### Files and Tags Which Should Be Monitored

The most important log file is *server.log*. However, all log files should be controlled. In the file *error.log*, only errors are reported, so this might also be a good hook for your log file monitoring.

All entries of type **ERROR** should be analyzed. Please take all possible use cases into consideration to configure which entries should really trigger a system alarm. For example, there might be log entries labeled as ERROR which occur rather often and only show that an access denied event has occurred. So at this point, the monitoring tool has to be adapted in a very specific way for each CM system.

## 12.8 Monitoring E-Mail Functionalities

---

### 12.8.1 CM Error E-Mail Configuration

The first step in monitoring e-mail functionalities of ConSol CM is configuring the correct TO addresses for error e-mails which concern the CM mailing subsystem. This is explained in detail in the *ConSol CM Administrator Manual*, section *Administrator and Notification E-Mail Addresses* (Appendix E in most editions).

Of course, you can configure TO addresses which belong to a ticketing system (e.g., ConSol CM, incident queue). In this way, you can be sure to get a notification when an e-mail error occurs.

### 12.8.2 Log File Control

Additionally, you can scan the specific log file, to be found in the following path:

- **JBoss 5:**

```
<JBoss_HOME>\server\<CMRF_SERVER_NAME>\log\mail.log
```

- **JBoss 7 (single server):**

```
<JBoss_HOME>/standalone/log/mail.log
```

- **Weblogic:**

```
<WLS_HOME>\Middleware\user_projects\domains\consolcm6_domain\cmrf-logs\mail.log
```

### 12.8.3 Control of Undeliverable E-Mails

E-mails which were fetched by CM but could not be processed further, are stored in the CM system:

- **NIMH mode**

In the database, table *cmas\_nimh\_archived\_mail*

- **Mule/ESB mode**

In the directory *<CMAS\_DATADIR>/mail/unparsable*

In the Admin Tool, those e-mail backups are listed under navigation group *E-Mail*, navigation item *E-Mail Backups*. In a well-maintained system, this table should be empty.

A detailed explanation is provided in the *ConSol CM Administrator Manual*, section *E-Mail Backups*.

## 12.9 CMRF/DWH Monitoring

---

### How can I monitor the CMRF function and DWH status?

You can, e.g.,

- check the value of *hlp\_parameter.dwh\_status* (explanation see section [hlp\\_parameter](#)). Should be 4 (*DWH operational*).
- check the *cmrf.log* file for entries which contain *ERROR* (maybe also *WARNING*)

The CM database (table *cmas\_dwh\_synchronization*) and the log files (*server.log*, *cmrf.log*) are the best source of information about status and progress.

- CM database:
  - the table *cmas\_dwh\_synchronization* contains information about each operation.
    - the column *dwh\_status* contains the status of the processing in CM6:
      - NEW - first status after creation
      - ACTIVE - data is sent to CMRF
      - SUCCESS - sending of data to CMRF has been finished successfully
      - ERROR - sending of data to CMRF has been finished unsuccessfully
    - Column *cmrf\_status* contains the status of the processing in CMRF:
      - NEW - first status after creation
      - ACTIVE - data is processed in CMRF
      - SUCCESS - processing of data has been finished successfully
      - ERROR - processing of data has been finished unsuccessfully
- Log files:
  - *server.log* (search for: *dwh-transfer-service*, *dwh-live-service*, *dwh-log-service*)
  - *cmrf.log*

## 13 ConSol CM Release Management Process: Test - Staging - Production

---

- [Introduction](#)
- [Deployment Pipeline](#)
  - [Development Server \(DEV\)](#)
  - [Test Server \(TEST\)](#)
  - [Staging Server \(STAGE\)](#)
  - [Production Server \(PROD\)](#)

## 13.1 Introduction

---

This section treats the deployment of a new ConSol CM configuration, not CM updates! For information about how to perform CM updates, please read section [System Update \(Minor Version\)](#).

Most larger ConSol CM systems are not only run on one server, but there are two or more servers, like:

- Development server (DEV)
- Test server (TEST)
- Staging server (STAGE)
- Production server (PROD)

In general IT infrastructure terminology, one of those servers is called a *tier*. Thus, the example above would describe the classical *4-tier deployment architecture*. Usually, the design and development of new custom-specific features and configurations is performed on the development (DEV) server and then moved to the test server (TEST) for QA. When the responsible persons have given their approval, the configuration can be moved to the staging server (STAGE) for complete tests and then be deployed in the production environment (PROD).

This implies that it is possible to move an entire CM configuration from one server to another one. For this purpose CM works with so called **scenarios**. A scenario contains the entire configuration of a CM system and can also contain runtime data. However, it is definitely not recommended to use a scenario to transport runtime data except for test environments. To be able to work correctly and efficiently with scenarios, you as a CM operator and/or administrator have to know the technical details about scenarios. They are explained in the *ConSol CM Administrator Manual* in section *Deployment (Import/Export)*. Please read this section, before you start work with scenario export/import! Here in this manual, we will assume that you know how to export and import scenarios and how to control exactly which data should be transferred!

## 13.2 Deployment Pipeline

---

Here, we assume, that only new functionalities should be implemented and no update will be performed. **All involved CM servers run exactly the same CM version!**

All servers (DEV-TEST-STAGE-PROD) are separate systems! No shared machines, databases, or file systems!



### GOLDEN RULE FOR CHANGES

When changes have to be made, do NEVER EVER perform those changes on STAGE or PROD! You might cause undesired side effects and might hamper the entire deployment, maintenance, and support processes!

A change always has to start on DEV (or in some cases on TEST). This includes major changes like the deployment of new workflows as well as minor changes like creating a new engineer account!

### 13.2.1 Development Server (DEV)

When new functionalities should be developed, new workflows should be implemented, or new configurations should be performed into the running CM system, a scenario from the production environment should be imported. Then new functionalities, configurations, and workflows can be implemented on this server. The server might even be a local machine of a developer or consultant.

Regarding the database and operating system: In practice they might differ from the middleware which is used in test-staging-production, e.g., as developer database, often MySQL is used even though in production Oracle or Microsoft SQL is used. Please be aware of the fact that a three-tier Java EE application might behave in a different way on different RDBMS. That is why we recommend to use identical RDBMS on DEV, TEST, STAGE, and PROD.

There are two ways to get back to the basic starting point in case you have reached a stage with your CM scenario which requires a new start:

1. Work with a VM and store a snapshot which you can use to start anew.
2. Work with the PROD scenario and import this again for a new start. In this case you have to start with an empty database, ask your CM consultant for help.

## 13.2.2 Test Server (TEST)

The test server has exactly the same parameters as the production server. A new deployment cycle starts with an exact copy (scenario) of the production server.



Make sure there are no e-mails sent from the TEST server to real persons except for the team involved in testing!

There are two ways of moving new content from DEV to TEST:

1. Export the scenario (configuration only, complete or partial, depending on the requirements) from DEV and import this scenario on TEST.
2. Implement exactly the same functionalities on TEST which you have implemented on DEV.

Then go ahead and test all functionalities, e.g., with the project stakeholders and with key users. There should always be a list of test cases, often derived from the list of use cases which have been defined in the requirements and specification process.

If all functionalities have been tested positive, the responsible persons give their approval and the scenario can be moved to staging.

During the tests on TEST, there might be some errors or some requirements might change. In those cases, the way goes back to DEV. When everything has been implemented in the new version, the DEV scenario can again be moved to TEST.



Make sure the mailboxes are configured correctly!! It is absolutely imperative to avoid that TEST and STAGE fetch e-mails from the same e-mail accounts.



### 13.2.3 Staging Server (STAGE)

The STAGE sever (sometimes also called *integration server*) is an identical copy of the PROD environment. The TEST scenario will be exported from TEST and imported on STAGE for final tests prior to the Go Live on PROD. All parameters and interfaces have to be exactly the same as in PROD. In this way, all CM core functionalities as well as system imports or other interactions with other IT systems can be tested. All changes which have been performed on TEST (e.g., implementation of ETL scripts, special flavors of CM as customer project, or specifically customized versions of CM.Track) have to be implemented on STAGE. The STAGE has to be tested under the same conditions as the PROD server will face, e.g., with the same number of concurrent users and the same amount of transferred data.

If this is possible, we recommend to transfer a snapshot of production data (via database export/import) into the STAGE CM database to be able to test with real data. This means:

- The scenario originates from the TEST environment.
- The data originates from the PROD environment.



Make sure there are no e-mails sent from the STAGE server to real persons except for the team involved in testing!

On STAGE, also performance tests have to be done. For example, it is possible that on TEST, your CM runs perfectly well, with only 10 users. When you move to STAGE, there are 100 users and a bottleneck in database access becomes obvious. This means: Not only functional but also non-functional requirements have to be taken into consideration on STAGE!

If everything on STAGE has been tested positive and the responsible persons have given their approval, the scenario can be exported for import on PROD.



Make sure the mailboxes are configured correctly!! It is absolutely imperative to avoid that STAGE and PROD fetch e-mails from the same e-mail accounts !!!

### 13.2.4 Production Server (PROD)



A deployment of a new CM scenario will imply a certain downtime of the system. Plan the CM Go Live in advance in order to avoid undesired side-effects in your every-day business process.

The PROD environment, also called *live environment*, is the environment which is used to work with in real business life.

On Go Live day, the scenario from STAGE has to be imported on PROD and all other adaptations have to be performed, e.g., implementation of ETL scripts, special flavors of CM as customer project, or specifically customized versions of CM.Track.

## 14 Troubleshooting FAQs

---

- [Introduction](#)
- [General Checklist](#)
- [What to Provide for the CM Support Team](#)
- [Some Error Scenarios and Their Solution](#)
  - [Problems with Mailing](#)
    - [An E-Mail Has Been Sent, but No Ticket Is Created?](#)
  - [Problems with LDAP Authentication](#)
    - [The Engineers Cannot Log In?](#)
    - [Only One Engineer Cannot Log In?](#)
  - [Problems with Kerberos Authentication](#)
    - [The Engineers Cannot Log In?](#)
  - [Problems with CMRF/DWH](#)
    - [Users Who Receive or Use Reports Based on the DWH Complain That Fields Are Missing in the Reports?](#)
    - [The Transfer CM - DWH Does Not Work?](#)
  - [Problems with Web Clients \(End Users: Engineers, Customers\)](#)
  - [Problems with the Search / the Indexer](#)
    - [Checklist for Problems with the Indexer](#)
      - [Checking the Indexer Status](#)
    - [Recreating the Index](#)

## 14.1 Introduction

---

When you are reading this chapter, CM has a problem, presumably one which you could not solve right off hand. In order to help you perform systematic error analysis, a [checklist](#) is provided.

Also in this section, some cases which can appear in practice with CM are covered. Of course, there are a lot more possible scenarios in customer-specific systems. Please call the ConSol CM support team (+49-89-45841-150) in case you need more help than the one we can provide in this manual. To make the support process as fast and as efficient as possible, we recommend to prepare the files and information listed in section [What to Provide for the CM Support Team](#).

## 14.2 General Checklist

---

If a CM error occurs and you do not have any idea what could be the cause, you can follow the checklist. Please keep in mind that there are a lot of collaborating systems (e.g., mail server, database server) which can be the source of the error, since CM is integrated into the IT landscape and is no stand-alone system.

1. As a first step, always check the entries in the [log files](#). The `server.log` file reports everything which happens in CM and should be the first entry point for troubleshooting. Maybe the reason for the error becomes obvious from this file. Search for `ERROR` in the log file.
2. Check the application server:
  - a. Is the application server up and running? With enough RAM and CPU? If CM is performing badly, not enough RAM or CPU power (or the [respective configuration for the application server](#)) can be the cause.
  - b. Does the application server have enough disc space? The [Indexer](#) needs enough disc space to be able to work correctly. (You can also [quickly check the Indexer status](#)).
3. Check the database server:
  - a. Is the database server of the CM database up and running?
  - b. Can CM connect without errors and without timeout to the database server?
4. Check the mail server (if mailing problems have occurred)
  - a. Is the mail server up and running? Is the e-mail fetched by CM?  
In Mule/ESB mode, the CM mail services have to be restarted after a mail server restart or downtime. Please refer to the *ConSol CM Administrator Manual*, section *Services, CM Services*, to learn how to restart the relevant services. In NIMH mode, a CM restart is not required, CM will "(re-) connect" automatically to the mail server.
  - b. For explanations of some special error scenarios, please see section [Checklist for Problems with Mailing](#).
5. If LDAP authentication is involved: Check the LDAP server.
  - a. Is the LDAP server up and running? Has there be a change in directory structure on the LDAP server?
  - b. For explanations of some special error scenarios, please see section [Problems with LDAP Authentication](#).

## 14.3 What to Provide for the CM Support Team

---

If you call ConSol CM support, it is of great help and will speed up the support process if you prepare the following data:

1. An exact history and description of what was done.  
Were there changes of the CM configuration using the Admin Tool? Were there changes in the workflows? Is it a problem with end user operation of the Web Client (CM Web Client or CM.Track)? What exactly was done when the problem occurred? To help with providing valuable information, we recommend to write a CM administrator log where everything which has been changed is documented.
2. The log files.  
The most important file (in most cases) is the *server.log* file. You can also prepare a compressed archive (*.zip*, *.tar*, etc.) with all log files to send it to the ConSol support team.
3. Exact information about all applications which are used with the exact version (e.g., operating system, web browser, Java version, database system, application server flavor and version).

## 14.4 Some Error Scenarios and Their Solution

---

### 14.4.1 Problems with Mailing

#### An E-Mail Has Been Sent, but No Ticket Is Created?

- The person (or system) who (which) receives the mail error e-mails (configured in the CM system property *cmas-esb-mail*, *mail.process.error* for Mule/ESB or *cmas-nimh-extension*, *mail.error.to.address* for NIMH) should have received an e-mail with an excerpt of the *mail.log* file. This can provide hints for a quick analysis.
- Which protocol is used? POP? IMAP? Check if the e-mails on the mail server are marked as *read*. CM fetches only e-mails which are marked *unread*.
- Is there an e-mail backup? See *ConSol CM Administrator Manual*, section *E-Mail Backups* for explanations. If a script which processes the incoming e-mails (see *ConSol CM Administrator Manual*, section *Scripts of Type E-Mail*) does not work, CM can fetch the e-mail and stores it in the CM database, but cannot import it into a ticket. To find out why, you can, e.g.,
  - check the *mail.log* file
  - increase the debug level for mail logging (see section [ConSol CM Logging and Log Files](#)) and add some statements like the following into the script:

```
if (mailLog.isDebugEnabled()) {  
    mailLog.debug("This is my info about the incoming message or something else  
$msg")  
}
```

- check the *mail.log* file again, there should be a more verbose output.
- Check if the [Indexer](#) is running and if the status is *GREEN*. When a ticket should be created, in most cases the index is used to find the correct customer (contact/company) for the ticket. Thus, when the Indexer has problems, there might be also problems with creating new tickets even though there is no connection between the problems at the first glance. You can [quickly check the Indexer status](#).
- Check the CM system property *cmas-esb-mail*, *mail.polling.interval*. This defines the time intervals in which CM fetches e-mails from the mail server. Maybe it is just too early and a new ticket could not have been created yet?

## 14.4.2 Problems with LDAP Authentication

### The Engineers Cannot Log In?

Is a password required to contact the LDAP server? Check the login to the LDAP server with the correct port using an LDAP browser.

Have any LDAP parameters which are set as CM system properties changed during the last hour/day? Check if all parameters are still valid. For detailed information refer to the *ConSol CM Administrator Manual*, section *LDAP Authentication in the Web Client*.

### Only One Engineer Cannot Log In?

Has the password been changed?

Is the engineer still present in the engineer administration in CM?

## 14.4.3 Problems with Kerberos Authentication

Since Kerberos is a highly complex topic, only some basic checks can be recommended.

### The Engineers Cannot Log In?

Does the CM application server (e.g., JBoss) run as the system user which has been used for registering the CM service in the Windows domain (Kerberos realm)? The user has to be identical.

Do all servers in the domain use the same system time? Since Kerberos tickets are based - besides others - on time stamps, it is indispensable that all servers work with the same system time (no more than five minutes delta for most systems).

## 14.4.4 Problems with CMRF/DWH

### Users Who Receive or Use Reports Based on the DWH Complain That Fields Are Missing in the Reports?

Check if all fields which are expected to be in the DWH are annotated correctly: Is for all Custom Fields, Data Object Group Fields, and Resource Fields, which should be transferred to the DWH, the annotation *reportable* set to *true*?

### The Transfer CM - DWH Does Not Work?

Check the DWH transfer mode first. Up to CM version 6.9.3.x, two modes are possible, *JMS* and *DIRECT*. Please read the detailed explanation about the DWH transfer mode in the *ConSol CM Administrator Manual*, section *Data Warehouse (DWH) Management*.



If *JMS* mode is used there might be a problem with the JMS queues. Check the location and status of the queues using tools of the respective application server.

In *DIRECT* mode (available in CM Versions 6.8.5.0 and up), the information for the CMRF are written to the (DWH!) database, so if there is a problem with the database connection between CM and the DWH, this mechanism will not work.

## 14.4.5 Problems with Web Clients (End Users: Engineers, Customers)

Most problems with the CM Web Client might be tasks for the CM administrator(s). However, you as an CM operator might also receive notifications about problems when CM engineers cannot work with the Web Client. We recommend to ask the engineers to always provide the exact message which is displayed in the Web Client with the exact timestamp. This makes it easier to find the respective lines in the log files. Furthermore they should provide information about:

- which browser they use
- in which version
- on which operating system
- the Java version

Thus, in a very first step, you can check if the components are supported by the current CM version. The information is provided in the *ConSol CM System Requirements*.

## 14.4.6 Problems with the Search / the Indexer

The searches in the ConSol CM system are based on indices created by the CM Indexer. The indices are stored on the file system, in the data directory which has been configured during system set-up and which is stored in the CM system property *cmas-core-server, data.directory*.

A detailed introduction to the Indexer principle is provided in the *ConSol CM Set-Up Manual*, section *Indexer Configuration*. Please read also the *ConSol CM Administrator Manual*, section *Search Configuration and Indexer Management* in order to learn details about how to manage the CM Indexer.

Problems with the Indexer can show various symptoms, for example:

- The search does not work at all (e.g., you enter a word in the Quick Search and nothing happens).
- Incoming e-mails are not processed correctly (because in the e-mail scripts a search for existing contacts/companies is performed. If this does not work, the entire script fails).
- Newly created objects (e.g., contacts, tickets, ticket data) cannot be found in the search, even though the data fields have the annotation *field-indexed* (usually set to *transitive*).

In those cases, you should - as a first step - always check if the Indexer works correctly. If there are unsolvable problems, you can recreate (synchronize) the index completely. If the index has only to be repaired, please take a look at the section in the *ConSol CM Administrator Manual* to see how to manage this.

## Checklist for Problems with the Indexer

### Checking the Indexer Status

There are two ways to check the Indexer status. They are both aimed at finding out if the status is *GREEN*, *YELLOW*, or *RED*. This is the value of the CM system property *cmas-core-index-common, index.status*.

You can retrieve the value by one of the following methods:

- **Use the Admin Tool**  
In the navigation group *System*, navigation item *System Properties* you can check the value.
- **Use database access**  
In the table *cmas\_configuration* look for *property\_ = indexer.status* and check the column *value\_*.

### Recreating the Index

If the index is corrupt, the files have been destroyed, or there are unsolvable problems with the Indexer you can completely recreate the index using one of two methods:

- **Use the Admin Tool**  
Open the navigation group *Services*, navigation item *Index* and select *Synchronize index*.
- **Use tools which access the MBeans**  
For example, you can use tools like [Jolokia](#) or [twiddle](#) (e.g., for JBoss) or *JConsole* (for JBoss) for graphic access. The MBeans of the Indexer offer the method *recreateIndex* which can be used to synchronize the entire index for the respective type of objects:
  - **Engineers:**  
core.engineerIndexService
  - **Resources:**  
core.resourceIndexService
  - **Tickets:**  
core.ticketIndexService
  - **Units (i.e., contacts and companies):**  
core.unitIndexService



Synchronizing the index completely might take a while in a large CM system (up to several hours), so please make sure you have planned this task for a time when CM search does not have to be available!

## 15 System (Fine) Tuning

---

- [Java Tuning](#)
  - [Start the ConSol CM Server with More RAM Space](#)
- [CM Tuning](#)
  - [Change Duration of Web Client Sessions](#)
  - [Mailing](#)
    - [Mailbox Full?](#)
    - [Change Mail Polling Interval](#)
    - [Managing the Maximum Attachment Size](#)
  - [Search /Indexer](#)
    - [The CM Engineers Complain That the Search Is Extremely Slow?](#)
  - [Database-Related Parameters](#)

There might be situations where you are informed by CM users that the system does not perform as well as desired. In this section, we provide information about some parameters which you can use to fine-tune a CM system. Of course, every CM instance is a highly specific and unique system which is configured exactly for the requirements of your company. Therefore it is not possible to document every parameter for each and every use case. However, we hope you can find some important information in the current chapter and get some ideas about how to improve the performance of your ConSol CM system.

## 15.1 Java Tuning

---

### 15.1.1 Start the ConSol CM Server with More RAM Space

The CM server is too slow but the machine has enough RAM space? Make sure, CM can use enough of this space.

In order to modify the size of the JVM (Java Virtual Machine) memory allocation pool modify the respective line of the application server start in the configuration file of the application server.

#### Examples:

- **Windows with JBoss 5:**  
...\jboss-5.1.0.GA\bin\run.conf.bat
- **Windows with JBoss 7:**  
...\jboss-eap-6.2\bin\standalone.conf.bat
- **Windows with Oracle WebLogic:**  
...\user\_projects\domains\<your domain>\createAdminServerService.cmd

The flag *Xmx* specifies the maximum memory allocation pool for a Java Virtual Machine (JVM), while *Xms* specifies the initial memory allocation pool, e.g.:

#### Before:

```
set "JAVA_OPTS=-Xms500M -Xmx1G -XX:MaxPermSize=256M"
```

#### After:

```
set "JAVA_OPTS=-Xms1G -Xmx1G -XX:MaxPermSize=256M"
```

A detailed explanation of ConSol CM start scripts is provided in the *ConSol CM Set-Up Manual*.

A more complex example for an optimized application server (JBoss) using Java start options is the following:

```
JAVA_OPTS="-Xms3072m -Xmx3072m -XX:MaxPermSize=640m -XX:ReservedCodeCacheSize=150m -Dorg.jboss.  
resolver.warning=true  
-Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -XX:  
+UseParallelOldGC  
-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/home/hotline/heapdump -verbose:gc -XX:  
+PrintGCDateStamps"
```

### Explanation:

- Start the application server with initial heap size of approx. 3 MB.

```
-Xms3072m
```

- The maximum heap size for the application sever is also approx. 3 MB.

```
-Xmx3072m
```

- The maximum space for the permanent space where compiled Java classes, methods and other required objects are stored during operation and never de-allocated. If this space is too small, even after a full garbage collection, an out of memory error will be thrown and the JVM will crash. This is no longer supported in Java 8!

```
-XX:MaxPermSize=640m
```

- The maximum size of the code cache where native Java code is stored during operation. A good explanation is provided on the [Oracle Codecache Tuning page](#).

```
-XX:ReservedCodeCacheSize=150m
```

- If the following parameter is set to true, a warning is written into the log file if the protocol is not file:// or vfsfile:// for a reference to a dtd.

```
-Dorg.jboss.resolver.warning=true
```

- The value of this property represents the maximum interval (in milliseconds) that the Java RMI runtime will allow between garbage collections of the local heap (client). A complete reference to *rmi* properties is provided on the [Oracle rmi properties reference page](#).

```
-Dsun.rmi.dgc.client.gcInterval=3600000
```

- The value of this property represents the maximum interval (in milliseconds) that the Java RMI runtime will allow between garbage collections of the local heap (server). A complete reference to *rmi* properties is provided on the [Oracle rmi properties reference page](#).

```
-Dsun.rmi.dgc.server.gcInterval=3600000
```

- The following parameter influences the Java Garbage Collection. *UseParallelOldGC* uses a multithreaded garbage collector for both, the new and the old generations, which usually provides faster collections. Used on multi-core systems.

```
-XX:+UseParallelOldGC
```

- Configures the JVM to generate a heap dump when an allocation from the Java heap or the permanent generation does not work, because there is not enough space. The heap dump is stored in the indicated path. This can help during troubleshooting.

```
-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/mydumdirectory
```

- Activates garbage collector logging. Writes one line for each young generation *gc* and for each full *gc*. With the *PrintGCDateStamps* option, each line starts with the absolute timestamp. Alias: -XX:+PrintGC

```
-verbose:gc -XX:+PrintGCDateStamps
```



If you want to know more about Java Garbage Collection, you might want to look at this web site: [Java Garbage Collection Basics, By Oracle \(last checked September 2016\)](#).

## 15.2 CM Tuning

---

These are examples taken from the most common FAQs (Frequently Asked Questions) concerning system properties in our consulting team. Please find a complete list of all system properties with detailed explanations in the appendix of the *ConSol CM Administrator Manual*.

### 15.2.1 Change Duration of Web Client Sessions

Do the Web Client sessions terminate too quickly and users (CM engineers) complain about being logged out while they did not actively work with CM only for a short time?

You can modify the session interval, after which CM terminates the engineer sessions automatically by modifying the CM system property *cmas-core-server, server.session.timeout* (indicated in milliseconds).



Please keep in mind that ConSol CM works with concurrent user licenses. Every engineer session consumes one license, and each session which is not terminated might prevent another engineer from logging in if no more license is available.

### 15.2.2 Mailing

#### Mailbox Full?

Users complain that no e-mails are received? You have checked the mailbox on the mail server, but all messages are present in the correct in-box?

If a mailbox which is configured as in-box for ConSol CM (for details, please refer to the *ConSol CM Administrator Manual*, section *E-Mail*) contains a very large number of e-mails which are marked as *SEEN*, this might cause a problem. When fetching the new e-mails, CM first has to retrieve a list of all e-mails in the mailbox to filter *SEEN* and new e-mails in the second step. When the first check takes longer than the polling interval (system property *cmas-esb-mail, mail.polling.interval*), there might be a deadlock situation and no e-mails can be fetched anymore.

As a solution you can of course increase the polling interval, but we recommend to clean-up the mailbox instead. Do not keep too many *SEEN* messages in the mailbox if it is not absolutely indispensable.

#### Change Mail Polling Interval

Users complain that e-mails which have been received by the mail server and are located in the in-box are not transferred to CM in the required time?



You can modify the mail polling interval in Mule/ESB environments by decreasing the value of the CM system property *cmas-esb-mail*, *mail.polling.interval*.

In NIMH environments, please take the value of *cmas-nimh,queue.task.interval.seconds* into consideration. The value of this CM System Property defines the interval for the check of the main mail poller.

## Managing the Maximum Attachment Size

Did an e-mail come in and create a ticket but the file which was originally attached to the e-mail was not attached to the ticket?

Then the maximum size which is allowed for attachments might be too small. You can increase the value of the CM system property *cmas-core-server, attachment.max.size*. This is a validation property of the CM API. It controls the size of attachments at tickets, at units, and at resources. It also controls the size of incoming ⚠️ (not outgoing!) e-mail attachments in NIMH as well as in Mule/ESB mode.



Do not set the allowed attachment size too large! Especially when the CM system property *cmas-core-index-common, index.attachment* is set to *true*, the system performance might decrease, because the Indexer needs a lot of time to index all the very large attachments.

### 15.2.3 Search /Indexer

#### The CM Engineers Complain That the Search Is Extremely Slow?

You might increase the number of threads the Indexer is running by increasing the value of the CM system property *cmas-core-index-common, index.task.worker.threads*. The default value is *1*. If you increase the value, do not set the value to a number higher than the number of CPU cores of the machine.

Of course there should always be enough space for the indexes on the hard drive and there should be enough RAM for a smooth operation of the Indexer without too many swap operations.

### 15.2.4 Database-Related Parameters

The CM engineers complain that CM in general is very slow?

One bottleneck can be the database access. You might want to increase the number of database connections in the connection pool. You can modify the size of the pool by setting the minimum and maximum pool size in the configuration file

- **JBoss 5:**  
...\\jboss-5.1.0.GA\\server\\cmas\\deploy\\cmDb-ds.xml

- **JBoss 7:**

...\jboss-eap-6.2\standalone\configuration\cm6.xml

Look for the tags *<min-pool-size>* and *<max-pool-size>*, e.g.:

```
...  
  
<pool>  
  <min-pool-size>5</min-pool-size>  
  <max-pool-size>200</max-pool-size>  
  <prefill>true</prefill>  
</pool>  
  
...
```

- **WebLogic:**

Use the Administration Console to modify the database connection pools.

## 16 Running CM in a Cluster

---

- [Introduction](#)
- [Basic Tips for Troubleshooting in a CM Cluster](#)
  - [Node Lost in Cluster](#)

## 16.1 Introduction

ConSol CM can be operated as multi-server system in a cluster. The set-up of CM clusters is described in detail in the *ConSol CM Set-Up Manual*. In this section we assume, a CM cluster is up and running and you as an operator are responsible for keeping it that way.

The following graphic provides a short overview of a possible CM cluster architecture.

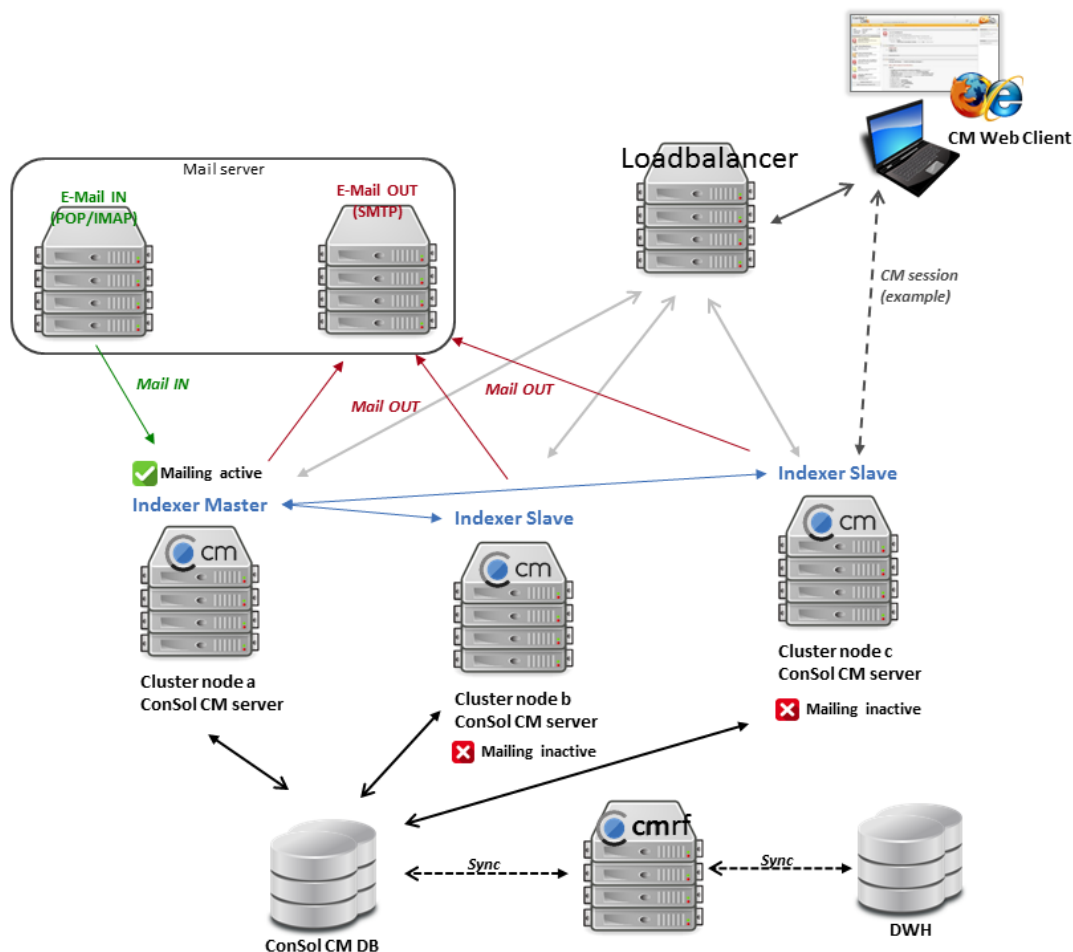


Fig. 1: Possible CM Cluster Architecture

Some basic principles apply to CM clusters:

- Every node runs the CM application.
- The Indexer runs as master indexing process on one server, all other servers are indexer slave servers. The Indexer principle is explained in detail in section [Indexer Management](#) in this manual.
- We recommend to run the CMRF (ConSol CM Reporting Framework) on a separate node. CMRF cannot be clustered. The CMRF set-up is described in the *ConSol CM Set-Up Manual*.

- Incoming e-mails can be fetched from the mail server by one CM node (as shown in the figure above) or (in NIMH mode) by all CM nodes. Please see the detailed explanation in the *ConSol CM Set-Up Manual*, section *E-Mail Configuration*.
- All CM nodes can send out e-mails (using SMTP).
- A load balancer can be used to distribute incoming requests between the CM nodes, but this is not mandatory. A CM Web Client will establish a session with one selected CM node.



There is no session failover, i.e., if a session between a CM client and a CM server node has crashed, a new session has to be established. The engineer has to log in again.

## 16.2 Basic Tips for Troubleshooting in a CM Cluster

---

### 16.2.1 Node Lost in Cluster

Check for the following patterns in the *server.log* file:

- `*INFO.*MyCM6Cluster.*Dead members:.*'`
- `*java.lang.IllegalStateException: Don.t know about node id.*'`

You might want to restart the respective node after having solved the problem.

## 17 Appendix A - Glossary

---

	Term	Explanation
A	Access Permissions	Permissions of an engineer to view or make changes to tickets in the Web Client. Access permissions are always assigned to a role, never to single engineers/users.
	ACIM	Activity item - entry in the history section of a ticket (e.g., comment, e-mail, attachment, time booking entry).
	Action Framework	ConSol CM module which enables CM to perform actions for specific objects. The Action Framework comprises Search Actions, Unit Actions and Resource Actions.
	AD	Microsoft Active Directory - an LDAP-based directory service for Microsoft Windows domain networks.
	Additional customer	Customer (contact or company) who is linked to a ticket besides the main customer, e.g., an employee of the customer's company. For additional customers, customer roles can be assigned.
	Admin Tool	ConSol CM component, graphical application to configure and manage a ConSol CM system. Uses Java Web Start.

	Term	Explanation
<b>B</b>	BI	Business Intelligence - methods, technologies, and architectures to transform data into useful information for business purposes.
<b>C</b>	CFEL	Custom Field Expression Language - Java classes and methods of the ConSol CM API to access data in Custom Fields and Data Object Group Fields.
	CMDB	ConSol CM Database - the working database of the CM system.
	CM.Doc	A standard module of ConSol CM which enables the engineer via ConSol CM Web Client to work with MS Word or Open Office documents pre-filled with ConSol CM ticket or customer parameters.
	CM.Phone	The ConSol CM module which provides CTI for CM.
	CM.Resource Pool	A ConSol CM module which is available in CM versions 6.10.1 and up. Enables CM to store objects of several types in the CM database as resources.
	CMRF	ConSol CM Reporting Framework - a JEE application which synchronizes data between the ConSol CM database and the DWH.
	CM.Track	ConSol CM web portal - provides customer access to the ConSol CM system.



	Term	Explanation
	Company	A data object of type <i>company</i> . Often this is a real company or an institution, but it can also be something else, like a machine or a ship.
	Contact	A data object of type <i>contact</i> . Often this is the person who has a question or service request, but it can also be something else, like a machine or a product.
	CTI	Computer Telephony Integration - a description for any technology that facilitates interaction between a telephone and a computer.
	Customer	General term for customer objects in ConSol CM. A customer can be a contact or a company. Technically, a customer is a data object. The respective Java class is <i>Unit</i> .
	Custom Field	A field where ticket data (e.g., priority, software module, etc.) can be stored.
	Custom Field Group	A group of Custom Fields where ticket data can be stored.
<b>D</b>	Data object	A customer (a contact or a company). Formerly <i>Unit</i> .
	Data Object Group	A group of fields where data for customers (contact or company) can be stored. Similar to Custom Field Group for ticket data.
	Data Object Group Field	A field where data for customers (contact or company) can be stored. Similar to Custom Field for ticket data.

	Term	Explanation
	DWH	Data Warehouse - ConSol CM database used for reporting and data analysis.
<b>E</b>	Engineer	User who has a login to the Web Client and who manages tasks defined in tickets.
	ERP system	Enterprise Resource Planning - often used for this type of enterprise management software.
	ESB	Enterprise Service Bus - a software architecture used for communication between mutually interacting software applications in a service-oriented architecture ( <a href="#">SOA</a> ).
	ETL	Extract Transform Load - extracts data from one source (a database or other source), transforms it, and loads it into a target system (e.g., another database)
<b>F</b>	FlexCDM	Flexible Customer Data Model - the customer data model introduced in ConSol CM in version 6.9. For each customer group, a specific customer data model can be defined.
<b>G</b>	GUI	Graphical User Interface
<b>I</b>	IMAP	Internet Message Access Protocol - Internet standard protocol to access e-mail on a remote e-mail server. Can be used as plain IMAP or as secure IMAP (IMAPs). In the latter case, proper certificates are required.
<b>J</b>	Java EE	Java Enterprise Edition

	Term	Explanation
	JMS	Java Message Service - Java EE component used to send messages between JMS clients.
	JRE	Java Runtime Environment. Provides a Java Virtual Machine for Clients.
<b>K</b>	Kerberos	A network authentication protocol based on (Kerberos) <i>tickets</i> which requires a special infrastructure.
	KPI	Key Performance Indicator - parameter used for performance measurement for companies, projects, etc.
<b>L</b>	LDAP	Lightweight Directory Access Protocol - application protocol to access and maintain directory information over an IP network.
	LDAPS	LDAP over SSL
<b>M</b>	Mailbox	Destination to which e-mail messages are delivered. Mailboxes are managed on an e-mail server. ConSol CM can access one or more mailboxes to retrieve e-mails.
	Main customer	The main customer of a ticket. Starting with ConSol CM version 6.9, this can be either a contact or a company.
	Mule	An open source Java-based Enterprise Service Bus (ESB).
<b>N</b>	NIMH	New Incoming Mail Handler - module for retrieving incoming e-mails, new in version 6.9.4.
<b>P</b>	PCDS	Page Customization Definition Section

	Term	Explanation
	Pentaho	Pentaho™ is a business intelligence (BI) suite which is available in open source and as enterprise editions.
	POP	Post Office Protocol - Internet standard protocol to retrieve e-mails from a remote server via TCP/IP. Can be used as plain POP or as secure POP (POPs). In the latter case, proper certificates are required.
	Portal	CM.Track - provides customer access to ConSol CM.
	Process Designer	ConSol CM component used to design, develop, and deploy workflows.
Q	Queue	Comprises tickets from the same domain and makes sure that all tickets of this domain are treated in the same way. A queue always has one workflow. Access rights and other parameters are defined based on queues.
R	RDBMS	Relational Database Management System - e.g. Oracle®, MS SQL Server®, MySQL.
	Resource	An object in a Resource Pool
	Resource Action	Part of the Action Framework. An Action based on a resource object.
	Resource Field	A field where resource data (e.g., inventory number, model, etc.) can be stored.

	Term	Explanation
	Resource Field Group	A group of fields where data for resources can be stored. Similar to Custom Field Group for ticket data.
	REST	Representational State Transfer - conventions for transferring data over HTTP connections.
	Role	Defines the access permissions and views of an engineer.
<b>S</b>	Script	Program written for a specific run-time environment that can interpret and automate the execution of tasks. In ConSol CM, scripts are stored in the Admin Tool and are stored as scripts for activities in workflows.
	Search Action	Part of the Action Framework. An Action which is based on the result set of a search.
	SMTP	Simple Message Transfer Protocol - standard protocol to send e-mails.
<b>T</b>	TAPI	Telephony Application Programming Interface - a Microsoft Windows API which provides computer/telephony integration and enables PCs running Microsoft Windows to use telephone services.
	TEF	Task Execution Framework - a ConSol CM module which can execute tasks asynchronously. A new feature as of version 6.9.4.
	Template	Preformatted example defining layout, text, and/or data for, e.g., e-mails or CM.Doc.

	Term	Explanation
	Ticket	Incident, service case, or other request of an internal or external customer. A ticket is the object which runs through the process defined by the workflow.
V	View	A selection of tickets based on scopes from one or more queues, assigned to a role and visible in the ticket list of the ConSol CM Web Client.
W	Workflow	Models a process that should be managed using ConSol CM step by step.

## 18 Appendix B - Trademarks

---

- Apache Lucene™ is a trademark of the Apache Software Foundation, see, e.g., the [Apache Software Foundation trademark policy page](#).
- JBoss® is a registered trademark of Red Hat Inc. Please see the [Red Hat trademark-guidelines-and-policies page](#).
- Microsoft® – Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Microsoft® Office – Microsoft and Microsoft Office are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Windows® operating system – Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Microsoft® Active Directory® – Microsoft and Microsoft Active Directory are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Microsoft® Word® – Microsoft and Microsoft Word are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Microsoft® SQL Server® – Microsoft and Microsoft SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- MuleSoft™ and Mule ESB™ are among the trademarks of MuleSoft, Inc. See [Mule Soft web page](#).
- Tomcat® is a registered trademark of the Apache Software Foundation. See [Apache Tomcat Legal page](#).
- Oracle® – Oracle is a registered trademark of Oracle Corporation and/or its affiliates. See [Oracle trademarks web page](#).
- Oracle® WebLogic – Oracle is a registered trademark of Oracle Corporation and/or its affiliates. See [Oracle trademarks web page](#).
- Pentaho® – Pentaho and the Pentaho logo are registered trademarks of Pentaho Inc. See [Pentaho trademark web page](#).

# Index

---

## A

Architecture of CM application [27](#)

## B

Backup of CM system [61](#)

boot.log [38](#)

## C

Cluster, running CM in a cluster [156](#)

CM Database [20](#)

cmrf.log [38](#)

CMRF (ConSol CM Reporting Framework) [23](#)

cmweb.log [38](#)

ctx.log [38](#)

## D

Data directory [29](#)

Data Warehouse management [90](#)

Development (DEV) server [135](#)

DWH (ConSol CM Data Warehouse) [23](#)

DWH management [90](#)

## E

E-mail, monitoring [130](#)

E-mail functionalities [70](#)

E-mail server [20](#)

E-mail tuning [73](#)

errors.log [38](#)

esb.log [38](#)

ex-reporter.log [38](#)

## F



FAQs, error scenarios and solutions [143](#)

File system structure [31](#)

## G

Glossary [159](#)

## I

IMAP [20](#)

index.log [38](#)

Indexer [21](#), [76](#)

Indexer, architecture, principle [79](#)

Indexer, backup and restore [85](#)

Indexer, monitoring [128](#)

Infrastructure, for CM Reporting [23](#)

## J

Java tuning (app server) [148](#)

## L

LDAP authentication [22](#), [109](#)

License management [113](#)

Log files [37](#)

Log files, configuration [41](#)

## M

mail.log [38](#)

Monitoring CM [122](#)

## O

Operationsmanual Consolcm [6](#), [6](#), [16](#), [18](#), [23](#), [27](#), [28](#), [37](#), [47](#), [60](#), [65](#), [69](#), [75](#), [89](#), [93](#), [108](#), [112](#), [121](#), [133](#), [139](#), [147](#), [155](#), [159](#), [167](#)

operationtimes.log [38](#)

## P

POP3 [20](#)

Production (PROD) server [138](#)

## R

Release process [134](#)

Restore of CM system [61](#)

## S

server.log [38](#)

session.log [38](#)

sql.log [39](#)

Staging (STAGE) server [137](#)

Startup, Shutdown of CM systems [48](#)

support\_libs\_errors.log [39](#)

System architecture [18](#)

System architecture with reporting [23](#)

## T

Test (TEST) server [136](#)

timer-manager.log [39](#)

Troubleshooting, checklist [141](#)

Tuning CM server [152](#)

tx.log [39](#)

## U

Update (minor version) of CM [67](#)

## W

workflow.log [39](#)